

Faster Parameterized Algorithms using Linear Programming*

Daniel Lokshтанov[†] N.S. Narayanaswamy[‡] Venkatesh Raman[§]
 M.S. Ramanujan[§] Saket Saurabh[§]

Abstract

We investigate the parameterized complexity of VERTEX COVER parameterized by the difference between the size of the optimal solution and the value of the linear programming (LP) relaxation of the problem. By carefully analyzing the change in the LP value in the branching steps, we argue that combining previously known preprocessing rules with the most straightforward branching algorithm yields an $O^*((2.618)^k)$ algorithm for the problem. Here k is the excess of the vertex cover size over the LP optimum, and we write $O^*(f(k))$ for a time complexity of the form $O(f(k)n^{O(1)})$, where $f(k)$ grows exponentially with k . We proceed to show that a more sophisticated branching algorithm achieves a runtime of $O^*(2.3146^k)$.

Following this, using known and new reductions, we give $O^*(2.3146^k)$ algorithms for the parameterized versions of ABOVE GUARANTEE VERTEX COVER, ODD CYCLE TRANSVERSAL, SPLIT VERTEX DELETION and ALMOST 2-SAT, and an $O^*(1.5214^k)$ algorithm for KOÑIG VERTEX DELETION, VERTEX COVER PARAM BY OCT and VERTEX COVER PARAM BY KVD. These algorithms significantly improve the best known bounds for these problems. The most notable improvement is the new bound for ODD CYCLE TRANSVERSAL - this is the first algorithm which beats the dependence on k of the seminal $O^*(3^k)$ algorithm of Reed, Smith and Vetta. Finally, using our algorithm, we obtain a kernel for the standard parameterization of VERTEX COVER with at most $2k - O(\log k)$ vertices. Our kernel is simpler than previously known kernels achieving the same size bound.

Topics: Algorithms and data structures. Graph Algorithms, Parameterized Algorithms.

1 Introduction and Motivation

In this paper we revisit one of the most studied problems in parameterized complexity, the VERTEX COVER problem. Given a graph $G = (V, E)$, a subset $S \subseteq V$ is called a *vertex cover* if every edge in E has at least one end-point in S . The VERTEX COVER problem is formally defined as follows.

*A preliminary version of this paper appears in the proceedings of STACS 2012.

[†]University of California San Diego, San Diego, USA. daniello@ii.uib.no

[‡]Department of CSE, IIT Madras, Chennai, India. swamy@cse.iitm.ernet.in

[§]The Institute of Mathematical Sciences, Chennai 600113, India.

[vraman|msramanujan|saket}@imsc.res.in](mailto:{vraman|msramanujan|saket}@imsc.res.in)

VERTEX COVER

Instance: An undirected graph G and a positive integer k .
Parameter: k .
Problem: Does G have a vertex cover of size at most k ?

We start with a few basic definitions regarding parameterized complexity. For decision problems with input size n , and a parameter k , the goal in parameterized complexity is to design an algorithm with runtime $f(k)n^{O(1)}$ where f is a function of k alone, as contrasted with a trivial $n^{k+O(1)}$ algorithm. Problems which admit such algorithms are said to be fixed parameter tractable (FPT). The theory of parameterized complexity was developed by Downey and Fellows [6]. For recent developments, see the book by Flum and Grohe [7].

VERTEX COVER was one of the first problems that was shown to be FPT [6]. After a long race, the current best algorithm for VERTEX COVER runs in time $O(1.2738^k + kn)$ [3]. However, when $k < m$, the size of the maximum matching, the VERTEX COVER problem is not interesting, as the answer is trivially NO. Hence, when m is large (for example when the graph has a perfect matching), the running time bound of the standard FPT algorithm is not practical, as k , in this case, is quite large. This led to the following natural “above guarantee” variant of the VERTEX COVER problem.

ABOVE GUARANTEE VERTEX COVER (AGVC)

Instance: An undirected graph G , a maximum matching M and a positive integer k .
Parameter: $k - |M|$.
Problem: Does G have a vertex cover of size at most k ?

In addition to being a natural parameterization of the classical VERTEX COVER problem, the AGVC problem has a central spot in the “zoo” of parameterized problems. We refer to Figure 1 for the details of problems reducing to AGVC. (See the Appendix for the definition of these problems.) In particular an improved algorithm for AGVC implies improved algorithms for several other problems as well, including ALMOST 2-SAT and ODD CYCLE TRANSVERSAL.

AGVC was first shown fixed parameter tractable by a parameter preserving reduction to ALMOST 2-SAT. In ALMOST 2-SAT, we are given a 2-SAT formula ϕ , a positive integer k and the objective is to check whether there exists at most k clauses whose deletion from ϕ can make the resulting formula satisfiable. The ALMOST 2-SAT problem was introduced in [16] and a decade later it was proved FPT by Razgon and O’Sullivan [23], who gave a $O^*(15^k)$ time algorithm for the problem. In 2011, there were two new algorithms for the AGVC problem [5, 22]. The first used new structural results about König-Egerváry graphs — graphs where the size of a minimum vertex cover is equal to the size of a maximum matching [22] while the second invoked a reduction to an “above guarantee version” of the MULTIWAY CUT problem [5]. The second algorithm runs in time $O^*(4^k)$ and this is also the fastest algorithm AGVC prior to our work.

In order to obtain the $O^*(4^k)$ running time bound for ABOVE GUARANTEE MULTIWAY CUT (and hence also for AGVC), Cygan et al [5] introduce a novel measure in terms of which

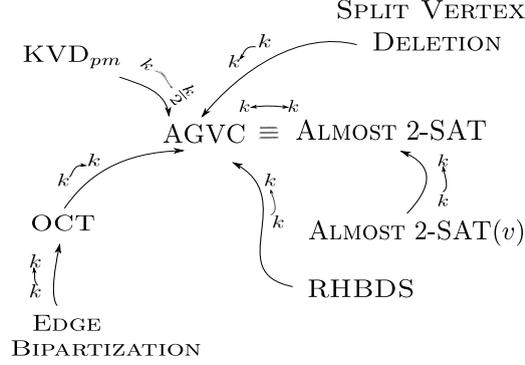


Figure 1: The zoo of problems around AGVC; An arrow from a problem P to a problem Q indicates that there is a parameterized reduction from P to Q with the parameter changes as indicated on the arrow.

the running time is bounded. Specifically they bound the running time of their algorithm in terms of the difference between the size of the solution the algorithm looks for and the value of the optimal solution to the linear programming relaxation of the problem. Since VERTEX COVER is a simpler problem than MULTIWAY CUT it is tempting to ask whether applying this new approach directly on VERTEX COVER could yield simpler and faster algorithms for AGVC. This idea is the starting point of our work.

The well known integer linear programming formulation (ILP) for VERTEX COVER is as follows.

ILP FORMULATION OF MINIMUM VERTEX COVER – ILPVC	
<i>Instance:</i>	A graph $G = (V, E)$.
<i>Feasible Solution:</i>	A function $x : V \rightarrow \{0, 1\}$ satisfying edge constraints $x(u) + x(v) \geq 1$ for each edge $(u, v) \in E$.
<i>Goal:</i>	To minimize $w(x) = \sum_{u \in V} x(u)$ over all feasible solutions x .

In the linear programming relaxation of the above ILP, the constraint $x(v) \in \{0, 1\}$ is replaced with $x(v) \geq 0$, for all $v \in V$. For a graph G , we call this relaxation LPVC(G). Clearly, every integer feasible solution is also a feasible solution to LPVC(G). If the minimum value of LPVC(G) is $vc^*(G)$ then clearly the size of a minimum vertex cover is at least $vc^*(G)$. This leads to the following parameterization of VERTEX COVER.

VERTEX COVER ABOVE LP	
<i>Instance:</i>	An undirected graph G , positive integers k and $\lceil vc^*(G) \rceil$, where $vc^*(G)$ is the minimum value of LPVC(G).
<i>Parameter:</i>	$k - \lceil vc^*(G) \rceil$.
<i>Problem:</i>	Does G have a vertex cover of of size at most k ?

Observe that since $vc^*(G) \geq m$, where m is the size of a maximum matching of G , we have that $k - vc^*(G) \leq k - m$. Thus, any parameterized algorithm for VERTEX COVER ABOVE

Problem Name	Previous $f(k)$ /Reference	New $f(k)$ in this paper
AGVC	4^k [5]	2.3146^k
ALMOST 2-SAT	4^k [5]	2.3146^k
RHORN-BACKDOOR DETECTION SET	4^k [5, 8]	2.3146^k
KÖNIG VERTEX DELETION	4^k [5, 18]	1.5214^k
SPLIT VERTEX DELETION	5^k [2]	2.3146^k
ODD CYCLE TRANSVERSAL	3^k [24]	2.3146^k
VERTEX COVER PARAM BY OCT	2^k (folklore)	1.5214^k
VERTEX COVER PARAM BY KVD	–	1.5214^k

Table 1: The table gives the previous $f(k)$ bound in the running time of various problems and the ones obtained in this paper.

LP is also a parameterized algorithm for AGVC and hence an algorithm for every problem depicted in Figure 1.

Our Results and Methodology. We develop a $O^*(2.3146^{(k-vc^*(G))})$ time branching algorithm for VERTEX COVER ABOVE LP. In an effort to present the key ideas of our algorithm in as clear a way as possible, we first present a simpler and slightly slower algorithm in Section 3. This algorithm exhaustively applies a collection of previously known preprocessing steps. If no further preprocessing is possible the algorithm simply selects an arbitrary vertex v and recursively tries to find a vertex cover of size at most k by considering whether v is in the solution or not. While the algorithm is simple, the analysis is more involved as it is not obvious that the measure $k - vc^*(G)$ actually drops in the recursive calls. In order to prove that the measure does drop we string together several known results about the linear programming relaxation of VERTEX COVER, such as the classical Nemhauser-Trotter theorem and properties of “minimum surplus sets”. We find it intriguing that, as our analysis shows, combining well-known reduction rules with naive branching yields fast FPT algorithms for all problems in Figure 1. We then show in Section 4 that adding several more involved branching rules to our algorithm yields an improved running time of $O^*(2.3146^{(k-vc^*(G))})$. Using this algorithm we obtain even faster algorithms for the problems in Figure 1.

We give a list of problems with their previous best running time and the ones obtained in this paper in Table 1. The most notable among them is the new algorithm for ODD CYCLE TRANSVERSAL, the problem of deleting at most k vertices to obtain a bipartite graph. The parameterized complexity of ODD CYCLE TRANSVERSAL was a long standing open problem in the area, and only in 2003 Reed et al. [24] developed an algorithm for the problem running in time $O^*(3^k)$. However, there has been no further improvement over this algorithm in the last 9 years; though several reinterpretations of the algorithm have been published [10, 15].

We also find the algorithm for KÖNIG VERTEX DELETION, the problem of deleting at most k vertices to obtain a König graph very interesting. KÖNIG VERTEX DELETION is a natural variant of the odd cycle transversal problem. In [18] it was shown that given a minimum vertex cover one can solve KÖNIG VERTEX DELETION in polynomial time. In this article we show a relationship between the measure $k - vc^*(G)$ and the minimum number of vertices needed to delete to obtain a König graph. This relationship together with a reduction rule

for KÖNIG VERTEX DELETION based on the Nemhauser-Trotter theorem gives an algorithm for the problem with running time $O^*(1.5124^k)$.

We also note that using our algorithm, we obtain a polynomial time algorithm for VERTEX COVER that, given an input (G, k) returns an equivalent instance $(G' = (V', E'), k')$ such that $k' \leq k$ and $|V(G')| \leq 2k - c \log k$ for any fixed constant c . This is known as a kernel for VERTEX COVER in the literature. We note that this kernel is simpler than another kernel with the same size bound [14].

We hope that this work will lead to a new race towards better algorithms for VERTEX COVER ABOVE LP like what we have seen for its classical counterpart, VERTEX COVER.

2 Preliminaries

For a graph $G = (V, E)$, for a subset S of V , the *subgraph of G induced by S* is denoted by $G[S]$ and it is defined as the subgraph of G with vertex set S and edge set $\{(u, v) \in E : u, v \in S\}$. By $N_G(u)$ we denote the (open) neighborhood of u , that is, the set of all vertices adjacent to u . Similarly, for a subset $T \subseteq V$, we define $N_G(T) = (\cup_{v \in T} N_G(v)) \setminus T$. When it is clear from the context, we drop the subscript G from the notation. We denote by $N_i[S]$, the set $N[N_{i-1}(S)]$ where $N_1[S] = N[S]$, that is, $N_i[S]$ is the set of vertices which are within a distance of i from a vertex in S . The surplus of an independent set $X \subseteq V$ is defined as $\mathbf{surplus}(X) = |N(X)| - |X|$. For a set \mathcal{A} of independent sets of a graph, $\mathbf{surplus}(\mathcal{A}) = \min_{X \in \mathcal{A}} \mathbf{surplus}(X)$. The surplus of a graph G , $\mathbf{surplus}(G)$, is defined to be the minimum surplus over all independent sets in the graph.

By the phrase an optimum solution to LPVC(G), we mean a feasible solution with $x(v) \geq 0$ for all $v \in V$ minimizing the objective function $w(x) = \sum_{u \in V} x(u)$. It is well known that for any graph G , there exists an optimum solution to LPVC(G), such that $x(u) \in \{0, \frac{1}{2}, 1\}$ for all $u \in V$ [19]. Such a feasible optimum solution to LPVC(G) is called a half integral solution and can be found in polynomial time [19]. In this paper we always deal with half integral optimum solutions to LPVC(G). Thus, by default whenever we refer to an *optimum solution* to LPVC(G) we will be referring to a *half integral optimum solution* to LPVC(G). Let $VC(G)$ be the set of all minimum vertex covers of G and $vc(G)$ denote the size of a minimum vertex cover of G . Let $VC^*(G)$ be the set of all optimal solutions (including non half integral optimal solution) to LPVC(G). By $vc^*(G)$ we denote the value of an optimum solution to LPVC(G). We define $V_i^x = \{u \in V : x(u) = i\}$ for each $i \in \{0, \frac{1}{2}, 1\}$ and define $x \equiv i$, $i \in \{0, \frac{1}{2}, 1\}$, if $x(u) = i$ for every $u \in V$. Clearly, $vc(G) \geq vc^*(G)$ and $vc^*(G) \leq \frac{|V|}{2}$ since $x \equiv \frac{1}{2}$ is always a feasible solution to LPVC(G). We also refer to the $x \equiv \frac{1}{2}$ solution simply as the all $\frac{1}{2}$ solution.

In branching algorithms, we say that a branching step results in a drop of (p_1, p_2, \dots, p_l) where $p_i, 1 \leq i \leq l$ is an integer, if the measure we use to analyze drops respectively by p_1, p_2, \dots, p_l in the corresponding branches. We also call the vector (p_1, p_2, \dots, p_l) the branching vector of the step.

3 A Simple Algorithm for VERTEX COVER ABOVE LP

In this section, we give a simpler algorithm for VERTEX COVER ABOVE LP. The algorithm has two phases, a preprocessing phase and a branching phase. We first describe the preprocessing steps used in the algorithm and then give a simple description of the algorithm. Finally, we argue about its correctness and prove the desired running time bound on the algorithm.

3.1 Preprocessing

We describe three standard preprocessing rules to simplify the input instance. We first state the (known) results which allow for their correctness, and then describe the rules.

Lemma 1. [20, 21] *For a graph G , in polynomial time, we can compute an optimal solution x to LPVC(G) such that all $\frac{1}{2}$ is the unique optimal solution to LPVC($G[V_{1/2}^x]$). Furthermore, $\text{surplus}(G[V_{1/2}^x]) > 0$.*

Lemma 2. [20] *Let G be a graph and x be an optimal solution to LPVC(G). There is a minimum vertex cover for G which contains all the vertices in V_1^x and none of the vertices in V_0^x .*

Preprocessing Rule 1. *Apply Lemma 1 to compute an optimal solution x to LPVC(G) such that all $\frac{1}{2}$ is the unique optimum solution to LPVC($G[V_{1/2}^x]$). Delete the vertices in $V_0^x \cup V_1^x$ from the graph after including V_1^x in the vertex cover we develop, and reduce k by $|V_1^x|$.*

In the discussions in the rest of the paper, we say that preprocessing rule 1 applies if all $\frac{1}{2}$ is not the unique solution to LPVC(G) and that it doesn't apply if all $\frac{1}{2}$ is the unique solution to LPVC(G).

The soundness/correctness of Preprocessing Rule 1 follows from Lemma 2. After the application of preprocessing rule 1, we know that $x \equiv \frac{1}{2}$ is the unique optimal solution to LPVC() of the resulting graph and the graph has a surplus of at least 1.

Lemma 3. [3, 20] *Let $G(V, E)$ be a graph, and let $S \subseteq V$ be an independent subset such that $\text{surplus}(Y) \geq \text{surplus}(S)$ for every set $Y \subseteq S$. Then there exists a minimum vertex cover for G , that contains all of S or none of S . In particular, if S is an independent set with the minimum surplus, then there exists a minimum vertex cover for G , that contains all of S or none of S .*

The following lemma, which handles without branching, the case when the minimum surplus of the graph is 1, follows from the above lemma.

Lemma 4. [3, 20] *Let G be a graph, and let $Z \subseteq V(G)$ be an independent set such that $\text{surplus}(Z) = 1$ and for every $Y \subseteq Z$, $\text{surplus}(Y) \geq \text{surplus}(Z)$. Then,*

1. *If the graph induced by $N(Z)$ is not an independent set, then there exists a minimum vertex cover in G that includes all of $N(Z)$ and excludes all of Z .*

2. If the graph induced by $N(Z)$ is an independent set, let G' be the graph obtained from G by removing $Z \cup N(Z)$ and adding a vertex z , followed by making z adjacent to every vertex $v \in G \setminus (Z \cup N(Z))$ which was adjacent to a vertex in $N(Z)$ (also called identifying the vertices of $N(Z)$). Then, G has a vertex cover of size at most k if and only if G' has a vertex cover of size at most $k - |Z|$.

We now give two preprocessing rules to handle the case when the surplus of the graph is 1.

Preprocessing Rule 2. If there is a set Z such that $\text{surplus}(Z) = 1$ and $N(Z)$ is not an independent set, then we apply Lemma 4 to reduce the instance as follows. Include $N(Z)$ in the vertex cover, delete $Z \cup N(Z)$ from the graph, and decrease k by $|N(Z)|$.

Preprocessing Rule 3. If there is a set Z such that $\text{surplus}(Z) = 1$ and the graph induced by $N(Z)$ is an independent set, then apply Lemma 4 to reduce the instance as follows. Remove Z from the graph, identify the vertices of $N(Z)$, and decrease k by $|Z|$.

The correctness of Preprocessing Rules 2 and 3 follows from Lemma 4. The entire preprocessing phase of the algorithm is summarized in Figure 2. Recall that each preprocessing rule can be applied only when none of the preceding rules are applicable, and that Preprocessing rule 1 is applicable if and only if there is a solution to $\text{LPVC}(G)$ which does not assign $\frac{1}{2}$ to every vertex. Hence, when Preprocessing Rule 1 does not apply all $\frac{1}{2}$ is the unique solution for $\text{LPVC}(G)$. We now show that we can test whether Preprocessing Rules 2 and 3 are applicable on the current instance in polynomial time.

Lemma 5. Given an instance (G, k) of VERTEX COVER ABOVE LP on which Preprocessing Rule 1 does not apply, we can test if Preprocessing Rule 2 applies on this instance in polynomial time.

Proof. We first prove the following claim.

Claim 1. The graph G (in the statement of the lemma) contains a set Z such that $\text{surplus}(G) = 1$ and $N(Z)$ is not independent if and only if there is an edge $(u, v) \in E$ such that solving $\text{LPVC}(G)$ with $x(u) = x(v) = 1$ results in a solution with value exactly $\frac{1}{2}$ greater than the value of the original $\text{LPVC}(G)$.

Proof. Suppose there is an edge (u, v) such that $w(x') = w(x) + \frac{1}{2}$ where x is the solution to the original $\text{LPVC}(G)$ and x' is the solution to $\text{LPVC}(G)$ with $x'(u) = x'(v) = 1$ and let $Z = V_0^{x'}$. We claim that the set Z is a set with surplus 1 and that $N(Z)$ is not independent. Since $N(Z)$ contains the vertices u and v , $N(Z)$ is not an independent set. Now, since $x \equiv \frac{1}{2}$ (Preprocessing Rule 1 does not apply), $w(x') = w(x) - \frac{1}{2}|Z| + \frac{1}{2}|N(Z)| = w(x) + \frac{1}{2}$. Hence, $|N(Z)| - |Z| = \text{surplus}(Z) = 1$.

Conversely, suppose that there is a set Z such that $\text{surplus}(Z) = 1$ and $N(Z)$ contains vertices u and v such that $(u, v) \in E$. Let x' be the assignment which assigns 0 to all vertices of Z , 1 to $N(Z)$ and $\frac{1}{2}$ to the rest of the vertices. Clearly, x' is a feasible assignment and $w(x') = |N(Z)| + \frac{1}{2}|V \setminus (Z \cup N(Z))|$. Since Preprocessing Rule 1 does not apply, $w(x') - w(x) = |N(Z)| - \frac{1}{2}(|Z| + |N(Z)|) = \frac{1}{2}(|N(Z)| - |Z|) = \frac{1}{2}$, which proves the converse part of the claim. □

Given the above claim, we check if Preprocessing Rule 2 applies by doing the following for every edge (u, v) in the graph.

Set $x(u) = x(v) = 1$ and solve the resulting LP looking for a solution whose optimum value is exactly $\frac{1}{2}$ more than the optimum value of LPVC(G).

□

Lemma 6. *Given an instance (G, k) of VERTEX COVER ABOVE LP on which Preprocessing Rules 1 and 2 do not apply, we can test if Preprocessing Rule 3 applies on this instance in polynomial time.*

Proof. We first prove a claim analogous to that proved in the above lemma.

Claim 2. *The graph G (in the statement of the lemma) contains a set Z such that $\mathbf{surplus}(G) = 1$ and $N(Z)$ is independent if and only if there is a vertex $u \in V$ such that solving LPVC(G) with $x(u) = 0$ results in a solution with value exactly $\frac{1}{2}$ greater than the value of the original LPVC(G).*

Proof. Suppose there is a vertex u such that $w(x') = w(x) + \frac{1}{2}$ where x is the solution to the original LPVC(G) and x' is the solution to LPVC(G) with $x'(u) = 0$ and let $Z = V_0^{x'}$. We claim that the set Z is a set with surplus 1 and that $N(Z)$ is independent. Since $x \equiv \frac{1}{2}$ (Preprocessing Rule 1 does not apply), $w(x') = w(x) - \frac{1}{2}|Z| + \frac{1}{2}|N(Z)| = w(x) + \frac{1}{2}$. Hence, $|N(Z)| - |Z| = \mathbf{surplus}(Z) = 1$. Since Preprocessing Rule 2 does not apply, it must be the case that $N(Z)$ is independent.

Conversely, suppose that there is a set Z such that $\mathbf{surplus}(Z) = 1$ and $N(Z)$ is independent. Let x' be the assignment which assigns 0 to all vertices of Z and 1 to all vertices of $N(Z)$ and $\frac{1}{2}$ to the rest of the vertices. Clearly, x' is a feasible assignment and $w(x') = |N(Z)| + \frac{1}{2}|V \setminus (Z \cup N(Z))|$. Since Preprocessing Rule 1 does not apply, $w(x') - w(x) = |N(Z)| - \frac{1}{2}(|Z| + |N(Z)|) = \frac{1}{2}(|N(Z)| - |Z|) = \frac{1}{2}$. This proves the converse part of the claim with u being any vertex of Z .

□

Given the above claim, we check if Preprocessing Rule 3 applies by doing the following for every vertex u in the graph.

Set $x(u) = 0$, solve the resulting LP and look for a solution whose optimum value exactly $\frac{1}{2}$ more than the optimum value of LPVC(G).

□

Definition 1. *For a graph G , we denote by $\mathcal{R}(G)$ the graph obtained after applying Preprocessing Rules 1, 2 and 3 exhaustively in this order.*

Strictly speaking $\mathcal{R}(G)$ is not a well defined function since the reduced graph could depend on which sets the reduction rules are applied on, and these sets, in turn, depend on the solution to the LP. To overcome this technicality we let $\mathcal{R}(G)$ be a function not only of the graph G but also of the representation of G in memory. Since our reduction rules are deterministic (and the LP solver we use as a black box is deterministic as well), running

The rules are applied in the order in which they are presented, that is, any rule is applied only when none of the earlier rules are applicable.

Preprocessing rule 1: Apply Lemma 1 to compute an optimal solution x to $\text{LPVC}(G)$ such that all $\frac{1}{2}$ is the unique optimum solution to $\text{LPVC}(G[V_{1/2}^x])$. Delete the vertices in $V_0^x \cup V_1^x$ from the graph after including V_1^x in the vertex cover we develop, and reduce k by $|V_1^x|$.

Preprocessing rule 2: Apply Lemma 5 to test if there is a set Z such that $\text{surplus}(Z) = 1$ and $N(Z)$ is not an independent set. If such a set does exist, then we apply Lemma 4 to reduce the instance as follows. Include $N(Z)$ in the vertex cover, delete $Z \cup N(Z)$ from the graph, and decrease k by $|N(Z)|$.

Preprocessing rule 3: Apply Lemma 6 to test if there is a set Z such that $\text{surplus}(Z) = 1$ and $N(Z)$ is an independent set. If there is such a set Z then apply Lemma 4 to reduce the instance as follows. Remove Z from the graph, identify the vertices of $N(Z)$, and decrease k by $|Z|$.

Figure 2: *Preprocessing Steps*

the reduction rules on (a specific representation of) G will always result in the same graph, making the function $\mathcal{R}(G)$ well defined. Finally, observe that for any G the all $\frac{1}{2}$ is the unique optimum solution to the $\text{LPVC}(\mathcal{R}(G))$ and $\mathcal{R}(G)$ has a surplus of at least 2.

3.2 Branching

After the preprocessing rules are applied exhaustively, we pick an arbitrary vertex u in the graph and branch on it. In other words, in one branch, we add u into the vertex cover, decrease k by 1, and delete u from the graph, and in the other branch, we add $N(u)$ into the vertex cover, decrease k by $|N(u)|$, and delete $\{u\} \cup N(u)$ from the graph. The correctness of this algorithm follows from the soundness of the preprocessing rules and the fact that the branching is exhaustive.

3.3 Analysis

In order to analyze the running time of our algorithm, we define a measure $\mu = \mu(G, k) = k - \text{vc}^*(G)$. We first show that our preprocessing rules do not increase this measure. Following this, we will prove a lower bound on the decrease in the measure occurring as a result of the branching, thus allowing us to bound the running time of the algorithm in terms of the measure μ . For each case, we let (G', k') be the instance resulting by the application of the rule or branch, and let x' be an optimum solution to $\text{LPVC}(G')$.

1. Consider the application of Preprocessing Rule 1. We know that $k' = k - |V_1^x|$. Since $x' \equiv \frac{1}{2}$ is the unique optimum solution to $\text{LPVC}(G')$, and G' comprises precisely the vertices of $V_{1/2}^x$, the value of the optimum solution to $\text{LPVC}(G')$ is exactly $|V_1^x|$ less than that of G . Hence, $\mu(G, k) = \mu(G', k')$.
2. We now consider the application of Preprocessing Rule 2. We know that $N(Z)$ was not independent. In this case, $k' = k - |N(Z)|$. We also know that $w(x') = \sum_{u \in V} x'(u) =$

$w(x) - \frac{1}{2}(|Z| + |N(Z)|) + \frac{1}{2}(|V_1^{x'}| - |V_0^{x'}|)$. Adding and subtracting $\frac{1}{2}(|N(Z)|)$, we get $w(x') = w(x) - |N(Z)| + \frac{1}{2}(|N(Z)| - |Z|) + \frac{1}{2}(|V_1^{x'}| - |V_0^{x'}|)$. But, $Z \cup V_0^{x'}$ is an independent set in G , and $N(Z \cup V_0^{x'}) = N(Z) \cup V_1^{x'}$ in G . Since $\text{surplus}(G) \geq 1$, $|N(Z \cup V_0^{x'})| - |Z \cup V_0^{x'}| \geq 1$. Hence, $w(x') = w(x) - |N(Z)| + \frac{1}{2}(|N(Z \cup V_0^{x'})| - |Z \cup V_0^{x'}|) \geq w(x) - |N(Z)| + \frac{1}{2}$. Thus, $\mu(G', k') \leq \mu(G, k) - \frac{1}{2}$.

3. We now consider the application of Preprocessing Rule 3. We know that $N(Z)$ was independent. In this case, $k' = k - |Z|$. We claim that $w(x') \geq w(x) - |Z|$. Suppose that this is not true. Then, it must be the case that $w(x') \leq w(x) - |Z| - \frac{1}{2}$. We will now consider three cases depending on the value $x'(z)$ where z is the vertex in G' resulting from the identification of $N(Z)$.

Case 1: $x'(z) = 1$. Now consider the following function $x'' : V \rightarrow \{0, \frac{1}{2}, 1\}$. For every vertex v in $G' \setminus \{z\}$, retain the value assigned by x' , that is $x''(v) = x'(v)$. For every vertex in $N(Z)$, assign 1 and for every vertex in Z , assign 0. Clearly this is a feasible solution. But now, $w(x'') = w(x') - 1 + |N(Z)| = w(x') - 1 + (|Z| + 1) \leq w(x) - \frac{1}{2}$. Hence, we have a feasible solution of value less than the optimum, which is a contradiction.

Case 2: $x'(z) = 0$. Now consider the following function $x'' : V \rightarrow \{0, \frac{1}{2}, 1\}$. For every vertex v in $G' \setminus \{z\}$, retain the value assigned by x' , that is $x''(v) = x'(v)$. For every vertex in Z , assign 1 and for every vertex in $N(Z)$, assign 0. Clearly this is a feasible solution. But now, $w(x'') = w(x') + |Z| \leq w(x) - \frac{1}{2}$. Hence, we have a feasible solution of value less than the optimum, which is a contradiction.

Case 3: $x'(z) = \frac{1}{2}$. Now consider the following function $x'' : V \rightarrow \{0, \frac{1}{2}, 1\}$. For every vertex v in $G' \setminus \{z\}$, retain the value assigned by x' , that is $x''(v) = x'(v)$. For every vertex in $Z \cup N(Z)$, assign $\frac{1}{2}$. Clearly this is a feasible solution. But now, $w(x'') = w(x') - \frac{1}{2} + \frac{1}{2}(|Z| + |N(Z)|) = w(x') - \frac{1}{2} + \frac{1}{2}(|Z| + |Z| + 1) \leq w(x) - \frac{1}{2}$. Hence, we have a feasible solution of value less than the optimum, which is a contradiction.

Hence, $w(x') \geq w(x) - |Z|$, which implies that $\mu(G', k') \leq \mu(G, k)$.

4. We now consider the branching step.

- (a) Consider the case when we pick u in the vertex cover. In this case, $k' = k - 1$. We claim that $w(x') \geq w(x) - \frac{1}{2}$. Suppose that this is not the case. Then, it must be the case that $w(x') \leq w(x) - 1$. Consider the following assignment $x'' : V \rightarrow \{0, \frac{1}{2}, 1\}$ to LPVC(G). For every vertex $v \in V \setminus \{u\}$, set $x''(v) = x'(v)$ and set $x''(u) = 1$. Now, x'' is clearly a feasible solution and has a value at most that of x . But this contradicts our assumption that $x \equiv \frac{1}{2}$ is the unique optimum solution to LPVC(G). Hence, $w(x') \geq w(x) - \frac{1}{2}$, which implies that $\mu(G', k') \leq \mu(G, k) - \frac{1}{2}$.
- (b) Consider the case when we don't pick u in the vertex cover. In this case, $k' = k - |N(u)|$. We know that $w(x') = w(x) - \frac{1}{2}(|\{u\}| + |N(u)|) + \frac{1}{2}(|V_1^{x'}| - |V_0^{x'}|)$. Adding and subtracting $\frac{1}{2}(|N(u)|)$, we get $w(x') = w(x) - |N(u)| - \frac{1}{2}(|\{u\}| - |N(u)|) + \frac{1}{2}(|V_1^{x'}| - |V_0^{x'}|)$. But, $\{u\} \cup V_0^{x'}$ is an independent set in G , and $N(\{u\} \cup V_0^{x'}) = N(u) \cup V_1^{x'}$ in G . Since $\text{surplus}(G) \geq 2$, $|N(\{u\} \cup V_0^{x'})| - |\{u\} \cup V_0^{x'}| \geq 2$. Hence, $w(x') = w(x) - |N(u)| + \frac{1}{2}(|N(\{u\} \cup V_0^{x'})| - |\{u\} \cup V_0^{x'}|) \geq w(x) - |N(u)| + 1$. Hence, $\mu(G', k') \leq \mu(G, k) - 1$.

We have thus shown that the preprocessing rules do not increase the measure $\mu = \mu(G, k)$ and the branching step results in a $(\frac{1}{2}, 1)$ branching vector, resulting in the recurrence $T(\mu) \leq T(\mu - \frac{1}{2}) + T(\mu - 1)$ which solves to $(2.6181)^\mu = (2.6181)^{k-vc^*(G)}$. Thus, we get a $(2.6181)^{(k-vc^*(G))}$ algorithm for VERTEX COVER ABOVE LP.

Theorem 1. VERTEX COVER ABOVE LP can be solved in time $O^*((2.6181)^{k-vc^*(G)})$.

By applying the above theorem iteratively for increasing values of k , we can compute a minimum vertex cover of G and hence we have the following corollary.

Corollary 1. There is an algorithm that, given a graph G , computes a minimum vertex cover of G in time $O^*(2.6181^{(vc(G)-vc^*(G))})$.

4 Improved Algorithm for VERTEX COVER ABOVE LP

In this section we give an improved algorithm for VERTEX COVER ABOVE LP using some more branching steps based on the structure of the neighborhood of the vertex (set) on which we branch. The goal is to achieve branching vectors better than $(\frac{1}{2}, 1)$.

4.1 Some general claims to measure the drops

First, we capture the drop in the measure in the branching steps, including when we branch on a larger sized sets. In particular, when we branch on a set S of vertices, in one branch we set all vertices of S to 1, and in the other, we set all vertices of S to 0. Note, however that such a branching on S may not be exhaustive (as the branching doesn't explore the possibility that some vertices of S are set to 0 and some are set to 1) unless the set S satisfies the premise of Lemma 3. Let $\mu = \mu(G, k)$ be the measure as defined in the previous section.

Lemma 7. Let G be a graph with $\text{surplus}(G) = p$, and let S be an independent set. Let \mathcal{H}_S be the collection of all independent sets of G that contain S (including S). Then, including S in the vertex cover while branching leads to a decrease of $\min\{\frac{|S|}{2}, \frac{p}{2}\}$ in μ ; and the branching excluding S from the vertex cover leads to a drop of $\frac{\text{surplus}(\mathcal{H}_S)}{2} \geq \frac{p}{2}$ in μ .

Proof. Let (G', k') be the instance resulting from the branching, and let x' be an optimum solution to LPVC(G'). Consider the case when we pick S in the vertex cover. In this case, $k' = k - |S|$. We know that $w(x') = w(x) - \frac{|S|}{2} + \frac{1}{2}(|V_1^{x'}| - |V_0^{x'}|)$. If $V_0^{x'} = \emptyset$, then we know that $V_1^{x'} = \emptyset$, and hence we have that $w(x') = w(x) - \frac{|S|}{2}$. Else, by adding and subtracting $\frac{1}{2}(|S|)$, we get $w(x') = w(x) - |S| + \frac{|S|}{2} + \frac{1}{2}(|V_1^{x'}| - |V_0^{x'}|)$. However, $N(V_0^{x'}) \subseteq S \cup V_1^{x'}$ in G . Thus, $w(x') \geq w(x) - |S| + \frac{1}{2}(|N(V_0^{x'})| - |V_0^{x'}|)$. We also know that $V_0^{x'}$ is an independent set in G , and thus, $|N(V_0^{x'})| - |V_0^{x'}| \geq \text{surplus}(G) = p$. Hence, in the first case $\mu(G', k') \leq \mu(G, k) - \frac{|S|}{2}$ and in the second case $\mu(G', k') \leq \mu(G, k) - \frac{p}{2}$. Thus, the drop in the measure when S is included in the vertex cover is at least $\min\{\frac{|S|}{2}, \frac{p}{2}\}$.

Consider the case when we don't pick S in the vertex cover. In this case, $k' = k - |N(S)|$. We know that $w(x') = w(x) - \frac{1}{2}(|S| + |N(S)|) + \frac{1}{2}(|V_1^{x'}| - |V_0^{x'}|)$. Adding and subtracting $\frac{1}{2}(|N(S)|)$, we get $w(x') = w(x) - |N(S)| + \frac{1}{2}(|N(S)| - |S|) + \frac{1}{2}(|V_1^{x'}| - |V_0^{x'}|)$. But, $S \cup V_0^{x'}$ is an independent set in G , and $N(S \cup V_0^{x'}) = N(S) \cup V_1^{x'}$ in G . Thus, $|N(S \cup V_0^{x'})| -$

$|S \cup V_0^{x'}| \geq \mathbf{surplus}(\mathcal{H}_S)$. Hence, $w(x') = w(x) - |N(S)| + \frac{1}{2}(|N(S \cup V_0^{x'})| - |S \cup V_0^{x'}|) \geq w(x) - |N(S)| + \frac{\mathbf{surplus}(\mathcal{H}_S)}{2}$. Hence, $\mu(G', k') \leq \mu(G, k) - \frac{\mathbf{surplus}(\mathcal{H}_S)}{2}$. \square

Thus, after the preprocessing steps (when the surplus of the graph is at least 2), suppose we manage to find (in polynomial time) a set $S \subseteq V$ such that

- $\mathbf{surplus}(G) = \mathbf{surplus}(S) = \mathbf{surplus}(\mathcal{H}_S)$,
- $|S| \geq 2$, and
- that the branching that sets all of S to 0 or all of S to 1 is exhaustive.

Then, Lemma 7 guarantees that branching on this set right away leads to a $(1, 1)$ branching vector. We now explore the cases in which such sets do exist. Note that the first condition above implies the third from the Lemma 3. First, we show that if there exists a set S such that $|S| \geq 2$ and $\mathbf{surplus}(G) = \mathbf{surplus}(S)$, then we can find such a set in polynomial time.

Lemma 8. *Let G be a graph on which Preprocessing Rule 1 does not apply (i.e. all $\frac{1}{2}$ is the unique solution to $\text{LPVC}(G)$). If G has an independent set S' such that $|S'| \geq 2$ and $\mathbf{surplus}(S') = \mathbf{surplus}(G)$, then in polynomial time we can find an independent set S such that $|S| \geq 2$ and $\mathbf{surplus}(S) = \mathbf{surplus}(G)$.*

Proof. By our assumption we know that G has an independent set S' such that $|S'| \geq 2$ and $\mathbf{surplus}(S') = \mathbf{surplus}(G)$. Let $u, v \in S'$. Let \mathcal{H} be the collection of all independent sets of G containing u and v . Let x be an optimal solution to $\text{LPVC}(G)$ obtained after setting $x(u) = 0$ and $x(v) = 0$. Take $S = V_0^x$, clearly, we have that $\{u, v\} \subseteq V_0^x$. We now have the following claim.

Claim 3. $\mathbf{surplus}(S) = \mathbf{surplus}(G)$.

Proof. We know that the objective value of $\text{LPVC}(G)$ after setting $x(u) = x(v) = 0$, $w(x) = |V|/2 + (|N(S)| - |S|)/2 = |V|/2 + \mathbf{surplus}(S)/2$, as all $\frac{1}{2}$ is the unique solution to $\text{LPVC}(G)$.

Another solution x' , for $\text{LPVC}(G)$ that sets u and v to 0, is obtained by setting $x'(a) = 0$ for every $a \in S'$, $x'(a) = 1$ for every $a \in N(S')$ and by setting all other variables to $1/2$. It is easy to see that such a solution is a feasible solution of the required kind and $w(x') = |V|/2 + (|N(S')| - |S'|)/2 = |V|/2 + \mathbf{surplus}(S')/2$. However, as x is also an optimum solution, $w(x) = w(x')$, and hence we have that $\mathbf{surplus}(S) \leq \mathbf{surplus}(S')$. But as S' is a set of minimum surplus in G , we have that $\mathbf{surplus}(S) = \mathbf{surplus}(S') = \mathbf{surplus}(G)$ proving the claim. \square

Thus, we can find a such a set S in polynomial time by solving $\text{LPVC}(G)$ after setting $x(u) = 0$ and $x(v) = 0$ for every pair of vertices u, v such that $(u, v) \notin E$ and picking that set V_0^x which has the minimum surplus among all x 's among all pairs u, v . Since any V_0^x contains at least 2 vertices, we have that $|S| \geq 2$. \square

4.2 (1,1) drops in the measure

Lemma 7 and Lemma 8 together imply that, if there is a minimum surplus set of size at least 2 in the graph, then we can find and branch on that set to get a (1, 1) drop in the measure.

Suppose that there is no minimum surplus set of size more than 1. Note that, by Lemma 7, when $\text{surplus}(G) \geq 2$, we get a drop of $(\text{surplus}(G))/2 \geq 1$ in the branch where we *exclude* a vertex or a set. Hence, if we find some vertex (set) to exclude in either branch of a two way branching, we get a (1, 1) branching vector. We now identify another such case.

Lemma 9. *Let v be a vertex such that $G[N(v) \setminus \{u\}]$ is a clique for some neighbor u of v . Then, there exists a minimum vertex cover that doesn't contain v or doesn't contain u .*

Proof. Towards the proof we first show the following well known observation.

Claim 4. *Let G be a graph and v be a vertex. Then there exists a minimum vertex cover for G containing $N(v)$ or at most $|N(v)| - 2$ vertices from $N(v)$.*

Proof. If a minimum vertex cover of G , say C , contains exactly $|N(v)| - 1$ vertices of $N(v)$, then we know that C must contain v . Observe that $C' = C \setminus \{v\} \cup N(v)$ is also a vertex cover of G of the same size as C . However, in this case, we have a minimum vertex cover containing $N(v)$. Thus, there exists a minimum vertex cover of G containing $N(v)$ or at most $|N(v)| - 2$ vertices from $N(v)$. \square

Let v be a vertex such that $G[N(v) \setminus \{u\}]$ is a clique. Then, branching on v would imply that in one branch we are excluding v from the vertex cover and in the other we are including v . Consider the branch where we include v in the vertex cover. Since $G[N(v) \setminus \{u\}]$ is a clique we have to pick at least $|N(v)| - 2$ vertices from $G[N(v) \setminus \{u\}]$. Hence, by Claim 4, we can assume that the vertex u is not part of the vertex cover. This completes the proof. \square

Next, in order to identify another case where we might obtain a (1, 1) branching vector, we first observe and capture the fact that when Preprocessing Rule 2 is applied, the measure $k - vc^*(G)$ actually drops by at least $\frac{1}{2}$ (as proved in item 2 of the analysis of the simple algorithm in Section 3.3).

Lemma 10. *Let (G, k) be the input instance and (G', k') be the instance obtained after applying Preprocessing Rule 2. Then, $\mu(G', k') \leq \mu(G, k) - \frac{1}{2}$.*

Thus, after we branch on an arbitrary vertex, if we are able to apply Preprocessing Rule 2 in the branch where we include that vertex, we get a (1, 1) drop. For, in the branch where we exclude the vertex, we get a drop of 1 by Lemma 7, and in the branch where we include the vertex, we get a drop of $\frac{1}{2}$ by Lemma 7, which is then followed by a drop of $\frac{1}{2}$ due to Lemma 10.

Thus, after preprocessing, the algorithm performs the following steps (see Figure 3) each of which results in a (1, 1) drop as argued before. Note that Preprocessing Rule 1 cannot apply in the graph $G \setminus \{v\}$ since the surplus of G can drop by at most 1 by deleting a vertex. Hence, checking if rule **B3** applies is equivalent to checking if, for some vertex v , Preprocessing Rule 2 applies in the graph $G \setminus \{v\}$. Recall that, by Lemma 5 we can check

this in polynomial time and hence we can check if **B3** applies on the graph in polynomial time.

<p>Branching Rules. These branching rules are applied in this order.</p> <p>B 1. Apply Lemma 8 to test if there is a set S such that $\mathbf{surplus}(S)=\mathbf{surplus}(G)$ and $S \geq 2$. If so, then branch on S.</p> <p>B 2. Let v be a vertex such that $G[N(v) \setminus \{u\}]$ is a clique for some vertex u in $N(v)$. Then in one branch add $N(v)$ into the vertex cover, decrease k by $N(v)$, and delete $N[v]$ from the graph. In the other branch add $N(u)$ into the vertex cover, decrease k by $N(u)$, and delete $N[u]$ from the graph.</p> <p>B 3. Apply Lemma 5 to test if there is a vertex v such that preprocessing Rule 2 applies in $G \setminus \{v\}$. If there is such a vertex, then branch on v.</p>
--

Figure 3: *Outline of the branching steps yielding (1, 1) drop.*

4.3 A Branching step yielding (1/2, 3/2) drop

Now, suppose none of the preprocessing and branching rules presented thus far apply. Let v be a vertex with degree at least 4. Let $S = \{v\}$ and recall that \mathcal{H}_S is the collection of all independent sets containing S , and $\mathbf{surplus}(\mathcal{H}_S)$ is an independent set with minimum surplus in \mathcal{H}_S . We claim that $\mathbf{surplus}(\mathcal{H}_S) \geq 3$.

As the preprocessing rules don't apply, clearly $\mathbf{surplus}(\mathcal{H}_S) \geq \mathbf{surplus}(G) \geq 2$. If $\mathbf{surplus}(\mathcal{H}_S) = 2$, then the set that realizes $\mathbf{surplus}(\mathcal{H}_S)$ is not S (as the $\mathbf{surplus}(S) = \text{degree}(v) - 1 = 3$), but a superset of S , which is of cardinality at least 2. Then, the branching rule **B1** would have applied which is a contradiction. This proves the claim. Hence by Lemma 7, we get a drop of at least 3/2 in the branch that excludes the vertex v resulting in a (1/2, 3/2) drop. This branching step is presented in Figure 4.

<p>B 4. If there exists a vertex v of degree at least 4 then branch on v.</p>
--

Figure 4: *The branching step yielding a (1/2, 3/2) drop.*

4.4 A Branching step yielding (1, 3/2, 3/2) drop

Next, we observe that when branching on a vertex, if in the branch that includes the vertex in the vertex cover (which guarantees a drop of 1/2), any of the branching rules **B1** or **B2** or **B3** applies, then combining the subsequent branching with this branch of the current branching step results in a net drop of (1, 3/2, 3/2) (which is (1, 1/2 + 1, 1/2 + 1)) (see Figure 5 (a)). Thus, we add the following branching rule to the algorithm (Figure 6).

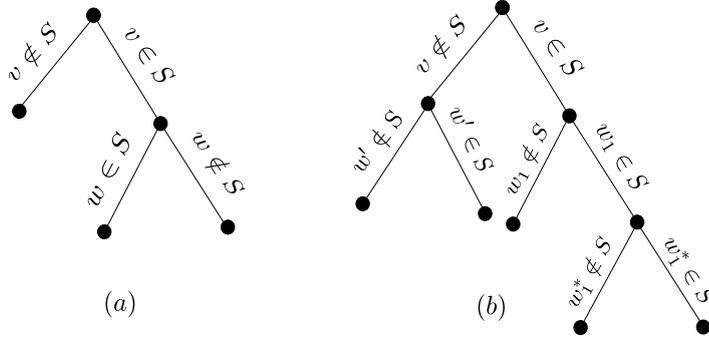


Figure 5: Illustrations of the branches of rules (a) **B5** and (b) **B6**

B 5. Let v be a vertex. If **B1** applies in $\mathcal{R}(G \setminus \{v\})$ or there exists a vertex w in $\mathcal{R}(G \setminus \{v\})$ on which either **B2** or **B3** applies then branch on v .

Figure 6: *The branching step yielding a $(1, 3/2, 3/2)$ drop.*

4.5 The Final branching step

Finally, if the preprocessing and the branching rules presented thus far do not apply, then note that we are left with a 3-regular graph. In this case, we simply pick a vertex v and branch. However, we execute the branching step more carefully in order to simplify the analysis of the drop. More precisely, we execute the following step at the end.

B 6. Pick an arbitrary degree 3 vertex v in G and let x, y and z be the neighbors of v . Then in one branch add v into the vertex cover, decrease k by 1, and delete v from the graph. The other branch that excludes v from the vertex cover, is performed as follows. Delete x from the graph, decrease k by 1, and obtain $\mathcal{R}(G \setminus \{x\})$. During the process of obtaining $\mathcal{R}(G \setminus \{x\})$, preprocessing rule 3 would have been applied on vertices y and z to obtain a ‘merged’ vertex v_{yz} (see proof of correctness of this rule). Now delete v_{yz} from the graph $\mathcal{R}(G \setminus \{x\})$, and decrease k by 1.

Figure 7: *Outline of the last step.*

4.6 Complete Algorithm and Correctness

A detailed outline of the algorithm is given in Figure 8. Note that we have already argued the correctness and analyzed the drops of all steps except the last step, **B6**.

The correctness of this branching rule will follow from the fact that $\mathcal{R}(G \setminus \{x\})$ is obtained by applying Preprocessing Rule 3 alone and that too only on the neighbors of x , that is, on the degree 2 vertices of $G \setminus \{x\}$ (Lemma 14). Lemma 18 (to appear later) shows the correctness of deleting v_{yz} from the graph $\mathcal{R}(G \setminus \{x\})$ without branching. Thus, the correctness of this algorithm follows from the soundness of the preprocessing rules and the fact that the

Preprocessing Step. Apply Preprocessing Rules 1, 2 and 3 in this order exhaustively on G .

Connected Components. Apply the algorithm on connected components of G separately. Furthermore, if a connected component has size at most 10, then solve the problem optimally in $O(1)$ time.

Branching Rules.
These branching rules are applied in this order.

B1 If there is a set S such that $\text{surplus}(S)=\text{surplus}(G)$ and $|S| \geq 2$, then branch on S .

B2 Let v be a vertex such that $G[N(v) \setminus \{u\}]$ is a clique for some vertex u in $N(v)$. Then in one branch add $N(v)$ into the vertex cover, decrease k by $|N(v)|$, and delete $N[v]$ from the graph. In the other branch add $N(u)$ into the vertex cover, decrease k by $|N(u)|$, and delete $N[u]$ from the graph.

B3 Let v be a vertex. If Preprocessing Rule 2 can be applied to obtain $\mathcal{R}(G \setminus \{v\})$ from $G \setminus \{v\}$, then branch on v .

B4 If there exists a vertex v of degree at least 4 then branch on v .

B5 Let v be a vertex. If **B1** applies in $\mathcal{R}(G \setminus \{v\})$ or if there exists a vertex w in $\mathcal{R}(G \setminus \{v\})$ on which **B2** or **B3** applies then branch on v .

B6 Pick an arbitrary degree 3 vertex v in G and let x, y and z be the neighbors of v . Then in one branch add v into the vertex cover, decrease k by 1, and delete v from the graph. The other branch, that excludes v from the vertex cover, is performed as follows. Delete x from the graph, decrease k by 1, and obtain $\mathcal{R}(G \setminus \{x\})$. Now, delete v_{yz} from the graph $\mathcal{R}(G \setminus \{x\})$, the vertex that has been created by the application of Preprocessing Rule 3 on v while obtaining $\mathcal{R}(G \setminus \{x\})$ and decrease k by 1.

Figure 8: *Outline of the Complete algorithm.*

branching is exhaustive.

The running time will be dominated by the way **B6** and the subsequent branching apply. We will see that **B6** is our most expensive branching rule. In fact, this step dominates the runtime of the algorithm of $O^*(2.3146^{\mu(G,k)})$ due to a branching vector of $(3/2, 3/2, 5/2, 5/2, 2)$. We will argue that when we apply **B6** on a vertex, say v , then on either side of the branch we will be able to branch using rules **B1**, or **B2**, or **B3** or **B4**. More precisely, we show that in the branch where we include v in the vertex cover,

- there is a vertex of degree 4 in $\mathcal{R}(G \setminus \{v\})$. Thus, **B4** will apply on the graph $\mathcal{R}(G \setminus \{v\})$ (if any of the earlier branching rules applied in this graph, then rule **B5** would have applied on G).
- $\mathcal{R}(G \setminus \{v\})$ has a degree 4 vertex w such that there is a vertex of degree 4 in the graph $\mathcal{R}(\mathcal{R}(G \setminus \{v\}) \setminus \{w\})$ and thus one of the branching rules **B1**, **B2**, **B3** or **B4** applies on the graph $\mathcal{R}(\mathcal{R}(G \setminus \{v\}) \setminus \{w\})$.

Rule	B1	B2	B3	B4	B5	B6
Branching Vector	(1,1)	(1,1)	(1,1)	$(\frac{1}{2}, \frac{3}{2})$	$(\frac{3}{2}, \frac{3}{2}, 1)$	$(\frac{3}{2}, \frac{3}{2}, \frac{5}{2}, \frac{5}{2}, 2)$
Running time	2^μ	2^μ	2^μ	2.1479^μ	2.3146^μ	2.3146^μ

Figure 9: A table giving the decrease in the measure due to each branching rule.

Similarly, in the branch where we exclude the vertex v from the solution (and add the vertices x and v_{yz} into the vertex cover), we will show that a degree 4 vertex remains in the reduced graph. This yields the claimed branching vector (see Figure 9). The rest of the section is geared towards showing this.

We start with the following definition.

Definition 2. *We say that a graph G is irreducible if Preprocessing Rules 1, 2 and 3 and the branching rules **B1**, **B2**, **B3**, **B4** and **B5** do not apply on G .*

Observe that when we apply **B6**, the current graph is 3-regular. Thus, after we delete a vertex v from the graph G and apply Preprocessing Rule 3 we will get a degree 4 vertex. Our goal is to identify conditions that ensure that the degree 4 vertices we obtain by applying Preprocessing Rule 3 survive in the graph $\mathcal{R}(G \setminus \{v\})$. We prove the existence of degree 4 vertices in subsequent branches after applying **B6** as follows.

- We do a closer study of the way Preprocessing Rules 1, 2 and 3 apply on $G \setminus \{v\}$ if Preprocessing Rules 1, 2 and 3 and the branching rules **B1**, **B2** and **B3** do not apply on G . Based on our observations, we prove some structural properties of the graph $\mathcal{R}(G \setminus \{v\})$, This is achieved by Lemma 14.
- Next, we show that Lemma 14, along with the fact that the graph is irreducible implies a lower bound of 7 on the length of the shortest cycle in the graph (Lemma 16). This lemma allows us to argue that when the preprocessing rules are applied, their effect is local.
- Finally, Lemmas 14 and 16 together ensure the presence of the required number of degree 4 vertices in the subsequent branching (Lemma 17).

4.6.1 Main Structural Lemmas: Lemmas 14 and 16

We start with some simple well known observations that we use repeatedly in this section. These observations follow from results in [20]. We give proofs for completeness.

Lemma 11. *Let G be an undirected graph, then the following are equivalent.*

- (1) *Preprocessing Rule 1 applies (i.e. All $\frac{1}{2}$ is not the unique solution to the LPVC(G).)*
- (2) *There exists an independent set I of G such that $\mathbf{surplus}(I) \leq 0$.*
- (3) *There exists an optimal solution x to LPVC(G) that assigns 0 to some vertex.*

Proof. (1) \implies (3): As we know that the optimum solution is half-integral, there exists an optimum solution that assigns 0 or 1 to some vertex. Suppose no vertex is assigned 0. Then, for any vertex which is assigned 1, its value can be reduced to $\frac{1}{2}$ maintaining feasibility (as all its neighbors have been assigned value $\geq \frac{1}{2}$) which is a contradiction to the optimality of the given solution.

(3) \implies (2): Let $I = V_0^x$, and suppose that $\mathbf{surplus}(I) > 0$. Then consider the solution x' that assigns $1/2$ to vertices in $I \cup N(I)$, retaining the value of x for the other vertices. Then x' is a feasible solution whose objective value $w(x')$ drops from $w(x)$ by $(|N(I)| - |I|)/2 = \mathbf{surplus}(I)/2 > 0$ which is a contradiction to the optimality of x .

(2) \implies (1): Setting all vertices in I to 0, all vertices in $N(I)$ to 1 and setting the remaining vertices to $\frac{1}{2}$ gives a feasible solution whose objective value is at most $|V|/2$, and hence all $\frac{1}{2}$ is not the unique solution to LPVC(G). \square

Lemma 12. *Let G be an undirected graph, then the following are equivalent.*

- (1) *Preprocessing Rule 1 or 2 or 3 applies.*
- (2) *There exists an independent set I such that $\mathbf{surplus}(I) \leq 1$.*
- (3) *There exists a vertex v such that an optimal solution x to LPVC($G \setminus \{v\}$) assigns 0 to some vertex.*

Proof. The fact that (1) and (2) are equivalent follows from the definition of the preprocessing rules and Lemma 11.

(3) \implies (2). By Lemma 11, there exists an independent set I in $G \setminus \{v\}$ whose surplus is at most 0. The same set will have surplus at most 1 in G .

(2) \implies (3). Let $v \in N(I)$. Then I is an independent set in $G \setminus \{v\}$ with surplus at most 0, and hence by Lemma 11, there exists an optimal solution to LPVC($G \setminus \{v\}$) that assigns 0 to some vertex. \square

We now prove an auxiliary lemma about the application of Preprocessing Rule 3 which will be useful in simplifying later proofs.

Lemma 13. *Let G be a graph and G_R be the graph obtained from G by applying Preprocessing Rule 3 on an independent set Z . Let z denote the newly added vertex corresponding to Z in G_R .*

1. *If G_R has an independent set I such that $\mathbf{surplus}(I) = p$, then G also has an independent set I' such that $\mathbf{surplus}(I') = p$ and $|I'| \geq |I|$.*
2. *Furthermore, if $z \in I \cup N(I)$ then $|I'| > |I|$.*

Proof. Let Z denote the minimum surplus independent set on which Preprocessing Rule 3 has been applied and z denote the newly added vertex. Observe that since Preprocessing Rule 3 applies on Z , we have that Z and $N(Z)$ are independent sets, $|N(Z)| = |Z| + 1$ and $|N(Z)| \geq 2$.

Let I be an independent set of G_R such that $\mathbf{surplus}(I) = p$.

- If both I and $N(I)$ do not contain z then we have that G has an independent set I such that $\mathbf{surplus}(I) = p$.
- Suppose $z \in I$. Then consider the following set: $I' := I \setminus \{z\} \cup N(Z)$. Notice that z represents $N(Z)$ and thus I do not have any neighbors of $N(Z)$. This implies that I' is an independent set in G . Now we will show that $\mathbf{surplus}(I') = p$. We know that $|N(Z)| = |Z| + 1$ and $N(I') = N(I) \cup Z$. Thus,

$$\begin{aligned}
|N(I')| - |I'| &= (|N(I)| + |Z|) - |I'| \\
&= (|N(I)| + |Z|) - (|I| - 1 + |N(Z)|) \\
&= (|N(I)| + |Z|) - (|I| + |Z|) \\
&= |N(I)| - |I| = \mathbf{surplus}(I) = p.
\end{aligned}$$

- Suppose $z \in N(I)$. Then consider the following set: $I' := I \cup Z$. Notice that z represents $N(Z)$ and since $z \notin I$ we have that I do not have any neighbors of Z . This implies that I' is an independent set in G . We show that $\mathbf{surplus}(I') = p$. We know that $|N(Z)| = |Z| + 1$. Thus,

$$\begin{aligned}
|N(I')| - |I'| &= (|N(I)| - 1 + |N(Z)|) - |I'| \\
&= (|N(I)| - 1 + |N(Z)|) - (|I| + |Z|) \\
&= (|N(I)| + |Z|) - (|I| + |Z|) \\
&= |N(I)| - |I| = \mathbf{surplus}(I) = p.
\end{aligned}$$

From the construction of I' , it is clear that $|I'| \geq |I|$ and if $z \in (I \cup N(I))$ then $|I'| > |I|$. This completes the proof. \square

We now give some definitions that will be useful in formulating the statement of the main structural lemma.

Definition 3. Let G be a graph and $\mathcal{P} = P_1, P_2, \dots, P_\ell$ be a sequence of exhaustive applications of Preprocessing Rules 1, 2 and 3 applied in this order on G to obtain G' . Let $\mathcal{P}_3 = P_a, P_b, \dots, P_t$ be the subsequence of \mathcal{P} restricted to Preprocessing Rule 3. Furthermore let Z_j , $j \in \{a, \dots, t\}$ denote the minimum surplus independent set corresponding to P_t on which the Preprocessing Rule 3 has been applied and z_j denote the newly added vertex (See Lemma 4). Let $Z^* = \{z_j \mid j \in \{a, \dots, t\}\}$ be the set of these newly added vertices.

- We say that an applications of Preprocessing Rule 3 is trivial if the minimum surplus independent set Z_j on which P_j is applied has size 1, that is, $|Z_j| = 1$.
- We say that all applications of Preprocessing Rule 3 are independent if for all $j \in \{a, \dots, t\}$, $N[Z_j] \cap Z^* = \emptyset$.

Essentially, independent applications of Preprocessing Rule 3 mean that the set on which the rule is applied, and all its neighbors are vertices in the original graph.

Next, we state and prove one of the main structural lemmas of this section.

Lemma 14. *Let $G = (V, E)$ be a graph on which Preprocessing Rules 1, 2 and 3 and the branching rules **B1**, **B2** and **B3** do not apply. Then for any vertex $v \in V$,*

1. *preprocessing Rules 1 and 2 have not been applied while obtaining $\mathcal{R}(G \setminus \{v\})$ from $G \setminus \{v\}$;*
2. *and all applications of the Preprocessing Rule 3 while obtaining $\mathcal{R}(G \setminus \{v\})$ from $G \setminus \{v\}$ are independent and trivial.*

Proof. Fix a vertex v . Let $G_0 = G \setminus \{v\}, G_1, \dots, G_t = \mathcal{R}(G \setminus \{v\})$ be a sequence of graphs obtained by applying Preprocessing Rules 1, 2 and 3 in this order to obtain the reduced graph $\mathcal{R}(G \setminus \{v\})$.

We first observe that Preprocessing Rule 2 never applies in obtaining $\mathcal{R}(G \setminus \{v\})$ from $G \setminus \{v\}$ since otherwise, **B3** would have applied on G . Next, we show that Preprocessing Rule 1 does not apply. Let q be the least integer such that Preprocessing Rule 1 applies on G_q and it does not apply to any graph $G_{q'}, q' < q$. Suppose that $q \geq 1$. Then, only Preprocessing Rule 3 has been applied on G_0, \dots, G_{q-1} . This implies that G_q has an independent set I_q such that $\mathbf{surplus}(I_q) \leq 0$. Then, by Lemma 13, G_{q-1} also has an independent set I'_q such that $\mathbf{surplus}(I'_q) \leq 0$ and thus by Lemma 11 Preprocessing Rule 1 applies to G_{q-1} . This contradicts the assumption that on G_{q-1} Preprocessing Rule 1 does not apply. Thus, we conclude that q must be zero. So, $G \setminus \{v\}$ has an independent set I_0 such that $\mathbf{surplus}(I_0) \leq 0$ in $G \setminus \{v\}$ and thus I_0 is an independent set in G such that $\mathbf{surplus}(I_0) \leq 1$ in G . By Lemma 12 this implies that either of Preprocessing Rules 1, 2 or 3 is applicable on G , a contradiction to the given assumption.

Now we show the second part of the lemma. By the first part we know that the G_i 's have been obtained by applications of Preprocessing Rule 3 alone. Let $Z_i, 0 \leq i \leq t-1$ be the sets in G_i on which Preprocessing Rule 3 has been applied. Let the newly added vertex corresponding to $N(Z_i)$ in this process be z'_i . We now make the following claim.

Claim 5. *For any $i \geq 0$, if G_i has an independent set I_i such that $\mathbf{surplus}(I_i) = 1$, then G has an independent set I such that $|I| \geq |I_i|$ and $\mathbf{surplus}(I) = 2$. Furthermore, if $(I_i \cup N(I_i)) \cap \{z_1, \dots, z_{i-1}\} \neq \emptyset$, then $|I| > |I_i|$.*

Proof. We prove the claim by induction on the length of the sequence of graphs. For the base case consider $q = 0$. Since Preprocessing Rules 1, 2, and 3 do not apply on G , we have that $\mathbf{surplus}(G) \geq 2$. Since I_0 is an independent set in $G \setminus \{v\}$ we have that I_0 is an independent set in G also. Furthermore since $\mathbf{surplus}(I_0) = 1$ in $G \setminus \{v\}$, we have that $\mathbf{surplus}(I_0) = 2$ in G , as $\mathbf{surplus}(G) \geq 2$. This implies that G has an independent set I_0 with $\mathbf{surplus}(I_0) = 2 = \mathbf{surplus}(G)$. Furthermore, since G_0 does not have any newly introduced vertices, the last assertion is vacuously true. Now let $q \geq 1$. Suppose that G_q has a set I_q and $\mathbf{surplus}(I_q) = 1$. Thus, by Lemma 13, G_{q-1} also has an independent set I'_q such that $|I'_q| \geq |I_q|$ and $\mathbf{surplus}(I'_q) = 1$. Now by the induction hypothesis, G has an independent set I such that $|I| \geq |I'_q| \geq |I_q|$ and $\mathbf{surplus}(I) = 2 = \mathbf{surplus}(G)$.

Next we consider the case when $(I_q \cup N(I_q)) \cap \{z'_1, \dots, z'_{q-1}\} \neq \emptyset$. If $z'_{q-1} \notin I_q \cup N(I_q)$ then we have that I_q is an independent set in G_{q-1} such that $(I_q \cup N(I_q)) \cap \{z'_1, \dots, z'_{q-2}\} \neq \emptyset$. Thus, by induction we have that G has an independent set I such that $|I| > |I_q|$ and

$\text{surplus}(I) = 2 = \text{surplus}(G)$. On the other hand, if $z'_{q-1} \in I_q \cup N(I_q)$ then by Lemma 13, we know that G_{q-1} has an independent set I'_q such that $|I'_q| > |I_q|$ and $\text{surplus}(I'_q) = 1$. Now by induction hypothesis we know that G has an independent set I such that $|I| \geq |I'_q| > |I_q|$ and $\text{surplus}(I) = 2 = \text{surplus}(G)$. This concludes the proof of the claim. \square

We first show that all the applications of Preprocessing Rule 3 are trivial. Claim 5 implies that if we have a non-trivial application of Preprocessing Rule 3 then it implies that G has an independent set I such that $|I| \geq 2$ and $\text{surplus}(I) = 2 = \text{surplus}(G)$. Then, **B1** would apply on G , a contradiction.

Finally, we show that all the applications of Preprocessing Rule 3 are independent. Let q be the least integer such that the application of Preprocessing Rule 3 on G_q is not independent. That is, the application of Preprocessing Rule 3 on $G_{q'}$, $q' < q$, is trivial and independent. Observe that $q \geq 1$. We already know that every application of Preprocessing Rule 3 is trivial. This implies that the set Z_q contains a single vertex. Let $Z_q = \{z_q\}$. Since the application of Preprocessing Rule 3 on Z_q is not independent we have that $(Z_q \cup N(Z_q)) \cap \{z'_1, \dots, z'_{q-1}\} \neq \emptyset$. We also know that $\text{surplus}(Z_q) = 1$ and thus by Claim 5 we have that G has an independent set I such that $|I| \geq 2 > |Z_q|$ and $\text{surplus}(I) = 2 = \text{surplus}(G)$. This implies that **B1** would apply on G , a contradiction. Hence, we conclude that all the applications of Preprocessing Rule 3 are independent. This proves the lemma. \square

Let $g(G)$ denote the girth of the graph, that is, the length of the smallest cycle in G . Our next goal of this section is to obtain a lower bound on the girth of an irreducible graph. Towards this, we first introduce the notion of an *untouched* vertex.

Definition 4. *We say that a vertex v is untouched by an application of Preprocessing Rule 2 or Preprocessing Rule 3, if $\{v\} \cap (Z \cup N(Z)) = \emptyset$, where Z is the set on which the rule is applied.*

We now prove an auxiliary lemma regarding the application of the preprocessing rules on graphs of a certain girth and following that, we will prove a lower bound on the girth of irreducible graphs.

Lemma 15. *Let G be a graph on which Preprocessing Rules 1, 2 and 3 and the branching rules **B1**, **B2**, **B3** do not apply and suppose that $g(G) \geq 5$. Then for any vertex $v \in V$, any vertex $x \notin N_2[v]$ is untouched by the preprocessing rules applied to obtain the graph $\mathcal{R}(G \setminus \{v\})$ from $G \setminus \{v\}$ and has the same degree as it does in G .*

Proof. Since the preprocessing rules do not apply in G , the minimum degree of G is at least 3 and since the graph G does not have cycles of length 3 or 4, for any vertex v , the neighbors of v are independent and there are no edges between vertices in the first and second neighborhood of v .

We know by Lemma 14 that only Preprocessing Rule 3 applies on the graph $G \setminus \{v\}$ and it applies only in a trivial and independent way. Let $U = \{u_1, \dots, u_t\}$ be the degree 3 neighbors of v in G and let D represent the set of the remaining (high degree) neighbors of v . Let P_1, \dots, P_l be the sequence of applications of Preprocessing Rule 3 on the graph $G \setminus \{v\}$,

let Z_i be the minimum surplus set corresponding to the application of P_i and let z_i be the new vertex created during the application of P_i .

We prove by induction on i , that

- the application P_i corresponds to a vertex $u_j \in U$,
- any vertex $x \notin N_2[v] \setminus D$ is untouched by this application, and
- after the application of P_i , the degree of $x \notin N_2[v]$ in the resulting graph is the same as that in G .

In the base case, $i = 1$. Clearly, the only vertices of degree 2 in the graph $G \setminus \{v\}$ are the degree 3 neighbors of v . Hence, the application P_1 corresponds to some $u_j \in U$. Since the graph G has girth at least 5, no vertex in D can lie in the set $\{u_j\} \cup N(u_j)$ and hence must be untouched by the application of P_1 . Since u_j is a neighbor of v , it is clear that the application of P_1 leaves any vertex disjoint from $N_2[v]$ untouched. Now, suppose that after the application of P_1 , a vertex w disjoint from $N_2[v] \setminus D$ has lost a degree. Then, it must be the case that the application of P_1 identified two of w 's neighbors, say w_1 and w_2 as the vertex z_1 . But since P_1 is applied on the vertex u_j , this implies the existence of a 4 cycle u_j, w_1, w, w_2 in G , which is a contradiction.

We assume as induction hypothesis that the claim holds for all i' such that $1 \leq i' < i$ for some $i > 1$. Now, consider the application of P_i . By Lemma 14, this application cannot be on any of the vertices created by the application of P_l ($l < i$), and by the induction hypothesis, after the application of P_{i-1} , any vertex disjoint from $N_2[v] \setminus D$ remains untouched and retains the degree (which is ≥ 3) it had in the original graph. Hence, the application of P_i must occur on some vertex $u_j \in U$. Now, suppose that a vertex w disjoint from $N_2[v] \setminus D$ has lost a degree. Then, it must be the case that P_i identified two of w 's neighbors say w_1 and w_2 as the vertex z_i . Since P_i is applied on the vertex u_j , this implies the existence of a 4 cycle u_j, w_1, w, w_2 in G , which is a contradiction. Finally, after the application of P_i , since no vertex outside $N_2[v] \setminus D$ has ever lost degree and they all had degree at least 3 to begin with, we cannot apply Preprocessing Rule 3 any further. This completes the proof of the claim.

Hence, after applying Preprocessing Rule 3 exhaustively on $G \setminus \{v\}$, any vertex disjoint from $N_2[v]$ is untouched and has the same degree as in the graph G . This completes the proof of the lemma. \square

Recall that the graph is irreducible if none of the preprocessing rules or branching rules **B1** through **B5** apply, i.e: the algorithm has reached **B6**.

Lemma 16. *Let G be a connected 3-regular irreducible graph with at least 11 vertices. Then, $g(G) \geq 7$.*

Proof. 1. Suppose G contains a triangle v_1, v_2, v_3 . Let v_4 be the remaining neighbor of v_1 . Now, $G[N(v_1) \setminus \{v_4\}]$ is a clique, which implies that branching rule **B2** applies and hence contradicts the irreducibility of G . Hence, $g(G) \geq 4$.

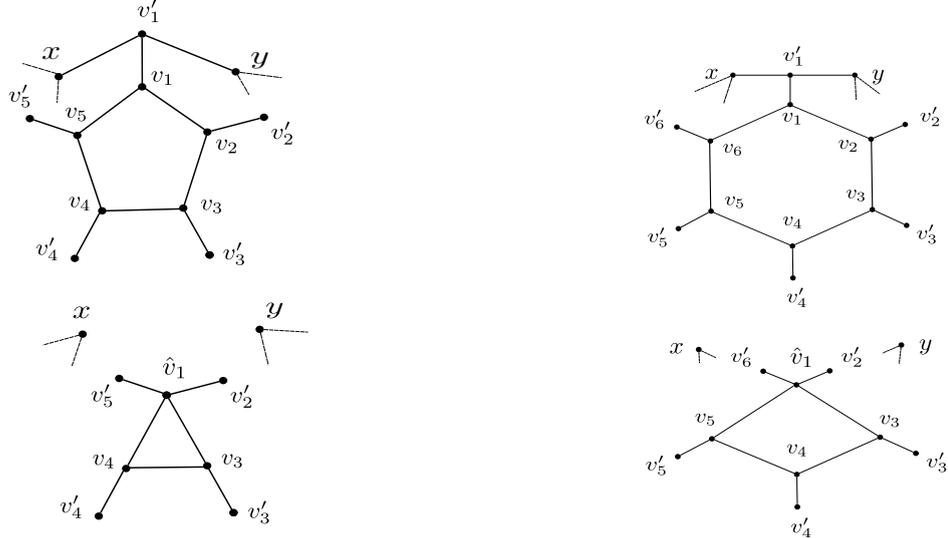


Figure 10: Cases of Lemma 16 when there is a 5 cycle or a 6 cycle in the graph

2. Suppose G contains a cycle v_1, v_2, v_3, v_4 of length 4. Since G does not contain triangles, it must be the case that v_1 and v_3 are independent. Recall that G has minimum surplus 2, and hence surplus of the set $\{v_1, v_3\}$ is at least 2. Since v_2 and v_4 account for two neighbors of both v_1 and v_3 , the neighborhood of $v_1 \cup v_3$ can contain at most 2 more vertices (G is 3 regular). Since the minimum surplus of G is 2, $|N(\{v_1, v_2\})| = 4$ and hence $\{v_1, v_3\}$ is a minimum surplus set of size 2, which implies that branching rule **B1** applies and hence contradicts the irreducibility of G . Hence, $g(G) \geq 5$.
3. Suppose that G contains a 5 cycle v_1, \dots, v_5 . Since $g(G) \geq 5$, this cycle does not contain chords. Let v'_i denote the remaining neighbor of the vertex v_i in the graph G . Since there are no triangles or 4 cycles, $v'_i \neq v'_j$ for any $i \neq j$, and for any i and j such that $|i - j| = 1$, v'_i and v'_j are independent. Now, we consider the following 2 cases.

Case 1: Suppose that for every i, j such that $|i - j| \neq 1$, v'_i and v'_j are adjacent. Then, since G is a connected 3-regular graph, G has size 10, which is a contradiction.

Case 2: Suppose that for some i, j such that $|i - j| \neq 1$, v'_i and v'_j are independent (see Figure 10). Assume without loss of generality that $i = 1$ and $j = 3$. Consider the vertex v'_1 and let x and y be the remaining 2 neighbors of v'_1 (the first neighbor being v_1). Note that x or y cannot be incident to v_3 , since otherwise x or y will coincide with v'_3 . Hence, v_3 is disjoint from $N_2[v'_1]$. By Lemma 14 and Lemma 15, only Preprocessing Rule 3 applies and the applications are only on the vertices v_1, x and y and leaves v_3 untouched and the degree of vertex v_3 unchanged. Now, let \hat{v}_1 be the vertex which is created as a result of applying Preprocessing Rule 3 on v_1 . Let \hat{v}_4 be the vertex created when v_4 is identified with another vertex during some application of Preprocessing Rule 3. If v_4 is untouched, then we let $\hat{v}_4 = v_4$. Similarly, let \hat{v}'_3 be the vertex created when v'_3 is identified with another vertex during some application of Preprocessing Rule 3. If v'_3 is untouched, then we let $\hat{v}'_3 = v'_3$. Since v_3 is untouched and its degree remains 3 in the graph $\mathcal{R}(G \setminus \{v\})$, the neighborhood of v_3 in this graph

can be covered by a 2 clique \hat{v}_1, \hat{v}_4 and a vertex \hat{v}'_3 , which implies that branching rule **B2** applies in this graph, implying that branching rule **B5** applies in the graph G , contradicting the irreducibility of G . Hence, $g(G) \geq 6$.

4. Suppose that G contains a 6 cycle v_1, \dots, v_6 . Since $g(G) \geq 6$, this cycle does not contain chords. Let v'_i denote the remaining neighbor of each vertex v_i in the graph G . Let x and y denote the remaining neighbors of v'_1 (see Figure 10). Note that both v_3 and v_5 are disjoint from $N_2[v'_1]$ (if this were not the case, then we would have cycles of length ≤ 5). Hence, by Lemma 14 and Lemma 15, we know that only Preprocessing Rule 3 applies and the applications are only on the vertices v_1, x and y , vertices v_3 and v_5 are untouched, and the degree of v_3 and v_5 in the graph $\mathcal{R}(G \setminus \{v'_1\})$ is 3. Let \hat{v}_1 be the vertex which is created as a result of applying P_3 on v_1 . Let \hat{v}_4 be the vertex created when v_4 is identified with another vertex during some application of P_3 . If v_4 is untouched, then we let $\hat{v}_4 = v_4$. Now, in the graph $\mathcal{R}(G \setminus \{v'_1\})$, the vertices v_3 and v_5 are independent and share two neighbors \hat{v}_1 and \hat{v}_4 . The fact that they have degree 3 each and the surplus of graph $\mathcal{R}(G \setminus \{v'_1\})$ is at least 2 (Lemma 14, Lemma 12) implies that $\{v_3, v_5\}$ is a minimum surplus set of size at least 2 in the graph $\mathcal{R}(G \setminus \{v'_1\})$, which implies that branching rule **B2** applies in this graph, implying that branching rule **B5** applies in the graph G , contradicting the irreducibility of G . Hence, $g(G) \geq 7$.

This completes the proof of the lemma. □

4.6.2 Correctness and Analysis of the last step

In this section we combine all the results proved above and show the existence of degree 4 vertices in subsequent branchings after **B6**. Towards this we prove the following lemma.

Lemma 17. *Let G be a connected 3 regular irreducible graph on at least 11 vertices. Then, for any vertex $v \in V$,*

1. $\mathcal{R}(G \setminus \{v\})$ contains three degree 4 vertices, say w_1, w_2, w_3 ; and
2. for any $w_i, i \in \{1, 2, 3\}$, $\mathcal{R}(\mathcal{R}(G \setminus \{v\}) \setminus \{w_i\})$ contains $w_j, i \neq j$ as a degree 4 vertex.

Proof. 1. Let v_1, v_2, v_3 be the neighbors of v . Since G was irreducible, **B1**, **B2**, **B3** do not apply on $\mathcal{R}(G \setminus \{v\})$ (else **B5** would have applied on G). By Lemma 14 and Lemma 15, we know that only Preprocessing Rule 3 would have applied and the applications are only on these three vertices. Let w_1, w_2 and w_3 be the three vertices which are created as a result of applying Preprocessing Rule 3 on these three vertices respectively. We claim that the degree of each w_i in the resulting graph is 4. Suppose that the degree of w_j is at most 3 for some j . But this can happen only if there was an edge between two vertices which are at a distance of 2 from v , that is, a path of length 3 between w_i and w_j for some $i \neq j$. This implies the existence of a cycle of length 5 in G , which contradicts Lemma 16.

2. Note that, by Lemma 15, it is sufficient to show that w_i is disjoint from $N_2[w_j]$ for any $i \neq j$. Suppose that this is not the case and let w_i lie in $N_2[w_j]$. First, suppose that w_i lies in $N_2[w_j] \setminus N_1[w_j]$ and there is no w_k in $N_1[w_i]$. Let x be a common neighbor of w_i

and w_j . This implies that, in G , x has paths of length 3 to v via w_i and via w_j , which implies the existence of a cycle of length at most 6, a contradiction. Now, suppose that w_i lies in $N_1[w_j]$. But this can happen only if there was an edge between two vertices which are at a distance of 2 from v . This implies the existence of a cycle of length 5 in G , contradicting Lemma 16. □

The next lemma shows the correctness of deleting v_{yz} from the graph $\mathcal{R}(G \setminus \{x\})$ without branching.

Lemma 18. *Let G be a connected irreducible graph on at least 11 vertices, v be a vertex of degree 3, and x, y, z be the set of its neighbors. Then, $G \setminus \{x\}$ contains a vertex cover of size at most k which excludes v if and only if $\mathcal{R}(G \setminus \{x\})$ contains a vertex cover of size at most $k - 3$ which contains v_{yz} , where v_{yz} is the vertex created in the graph $G \setminus \{x\}$ by the application of Preprocessing Rule 3 on the vertex v .*

Proof. We know by Lemma 15 that there will be exactly 3 applications of Preprocessing Rule 3 in the graph $G \setminus \{x\}$, and they will be on the three neighbors of x . Let G_1, G_2, G_3 be the graphs which result after each such application, in that order. We assume without loss of generality that the third application of Preprocessing Rule 3 is on the vertex v .

By the correctness of Preprocessing Rule 3, if $G \setminus \{x\}$ has a vertex cover of size at most k which excludes v , then G_2 has a vertex cover of size at most $k - 2$ which excludes v . Since this vertex cover must then contain y and z , it is easy to see that G_3 contains a vertex cover of size at most $k - 3$ containing v_{yz} .

Conversely, if G_3 has a vertex cover of size at most $k - 3$ containing v_{yz} , then replacing v_{yz} with the vertices y and z results in a vertex cover for G_2 of size at most $k - 2$ containing y and z (by the correctness of Preprocessing Rule 3). Again, by the correctness of Preprocessing Rule 3, it follows that $G \setminus \{x\}$ contains a vertex cover of size at most k containing y and z . Since v is adjacent to only y and z in $G \setminus \{x\}$, we may assume that this vertex cover excludes v . □

Thus, when branching rule **B6** applies on the graph G , we know the following about the graph.

- G is a 3 regular graph. This follows from the fact that Preprocessing Rules 1, 2 and 3 and the branching rule **B4** do not apply.
- $g(G) \geq 7$. This follows from Lemma 16.

Let v be an arbitrary vertex and x, y and z be the neighbors of v . Since G is irreducible, Lemma 17 implies that $\mathcal{R}(G \setminus \{x\})$ contains 3 degree 4 vertices, w_1, w_2 and w_3 . We let v_{yz} be w_1 . Lemma 17 also implies that for any i , the graph $\mathcal{R}(\mathcal{R}(G \setminus \{x\}) \setminus \{w_i\})$ contains 2 degree 4 vertices. Since the vertex v_{yz} is one of the three degree 4 vertices, in the graph $\mathcal{R}(\mathcal{R}(G \setminus \{x\}) \setminus v_{yz})$, the vertices w_2 and w_3 have degree 4 and one of the branching rules **B1**, or **B2**, or **B3** or **B4** will apply in this graph. Hence, we combine the execution of the rule **B6** along with the subsequent execution of one of the rules **B1**, **B2**, **B3** or **B4** (see Fig. 5).

To analyze the drops in the measure for the combined application of these rules, we consider each root to leaf path in the tree of Fig. 5 (b) and argue the drops in each path.

- Consider the subtree in which v is not picked in the vertex cover from G , that is, x is picked in the vertex cover, following which we branch on some vertex w during the subsequent branching, from the graph $\mathcal{R}(\mathcal{R}(G \setminus \{x\}) \setminus v_{yz})$.

Let the instances (corresponding to the nodes of the subtree) be (G, k) , (G_1, k_1) , (G_2, k_2) and (G'_2, k'_2) . That is, $G_1 = \mathcal{R}(\mathcal{R}(G \setminus \{x\}) \setminus \{v_{yz}\})$, $G'_2 = \mathcal{R}(G_1 \setminus \{w\})$ and $G_2 = \mathcal{R}(G_1 \setminus N[w])$.

By Lemma 7, we know that $\mu(G \setminus \{x\}, k-1) \leq \mu(G, k) - \frac{1}{2}$. This implies that $\mu(\mathcal{R}(G \setminus \{x\}), k') \leq \mu(G, k) - \frac{1}{2}$ where $(\mathcal{R}(G \setminus \{x\}), k')$ is the instance obtained by applying the preprocessing rules on $G \setminus \{x\}$.

By Lemma 7, we also know that including v_{yz} into the vertex cover will give a further drop of $\frac{1}{2}$. Hence, $\mu(\mathcal{R}(G \setminus \{x\}) \setminus \{v_{yz}\}, k' - 1) \leq \mu(G, k) - 1$. Applying further preprocessing will not increase the measure. Hence $\mu(G_1, k_1) \leq \mu(G, k) - 1$.

Now, when we branch on the vertex w in the next step, we know that we use one of the rules **B1**, **B2**, **B3** or **B4**. Hence, $\mu(G_2, k_2) \leq \mu(G_1, k_1) - \frac{3}{2}$ and $\mu(G'_2, k'_2) \leq \mu(G_1, k_1) - \frac{1}{2}$ (since **B4** gives the worst branching vector). But this implies that $\mu(G_2, k_2) \leq \mu(G, k) - \frac{5}{2}$ and $\mu(G'_2, k'_2) \leq \mu(G, k) - \frac{3}{2}$.

This completes the analysis of the branch of rule **B6** where v is not included in the vertex cover.

- Consider the subtree in which v is included in the vertex cover, by Lemma 17 we have that $\mathcal{R}(G \setminus \{v\})$ has exactly three degree 4 vertices, say w_1, w_2, w_3 and furthermore for any w_i , $i \in \{1, 2, 3\}$, $\mathcal{R}(\mathcal{R}(G \setminus \{v\}) \setminus \{w_i\})$ contains 2 degree 4 vertices. Since G is irreducible, we have that for any vertex v in G , the branching rules **B1**, **B2** and **B3** do not apply on the graph $\mathcal{R}(G \setminus \{v\})$. Thus, we know that in the branch where we include v in the vertex cover, the first branching rule that applies on the graph $\mathcal{R}(G \setminus \{v\})$ is **B4**. Without loss of generality, we assume that **B4** is applied on the vertex w_1 . Thus, in the branch where we include w_1 in the vertex cover, we know that $\mathcal{R}(\mathcal{R}(G \setminus \{v\}) \setminus \{w_1\})$ contains w_2 and w_3 as degree 4 vertices, This implies that in the graph $\mathcal{R}(\mathcal{R}(G \setminus \{v\}) \setminus \{w_1\})$ one of the branching rules **B1**, **B2**, **B3** or **B4** apply on a vertex w_1^* . Hence, we combine the execution of the rule **B6** along with the subsequent executions of **B4** and one of the rules **B1**, **B2**, **B3** or **B4** (see Fig. 5).

We let the instances corresponding to the nodes of this subtree be (G, k) , (G_1, k_1) , (G_2, k_2) , (G'_2, k'_2) , (G_3, k_3) and (G'_3, k'_3) , where $G_1 = \mathcal{R}(G \setminus \{v\})$, $G_2 = \mathcal{R}(G_1 \setminus N[w_1])$, $G'_2 = \mathcal{R}(G_1 \setminus \{w_1\})$, $G_3 = \mathcal{R}(G'_2 \setminus N[w_1^*])$ and $G'_3 = \mathcal{R}(G'_2 \setminus \{w_1^*\})$.

Lemma 7, and the fact that preprocessing rules do not increase the measure implies that $\mu(G_1, k_1) \leq \mu(G, k)$.

Now, since **B4** has been applied to branch on w_1 , the analysis of the drop of measure due to **B4** shows that $\mu(G_2, k_2) \leq \mu(G_1, k_1) - \frac{3}{2}$ and $\mu(G'_2, k'_2) \leq \mu(G_1, k_1) - \frac{1}{2}$. Similarly, since, in the graph G'_2 , we branch on vertex w_1^* using one of the rules **B1**, **B2**, **B3** or **B4**, we have that $\mu(G_3, k_3) \leq \mu(G'_2, k'_2) - \frac{3}{2}$ and $\mu(G'_3, k'_3) \leq \mu(G'_2, k'_2) - \frac{1}{2}$.

Combining these, we get that $\mu(G_3, k_3) \leq \mu(G, k) - \frac{5}{2}$ and $\mu(G'_3, k'_3) \leq \mu(G, k) - \frac{3}{2}$. This completes the analysis of rule **B6** where v is included in the vertex cover. Combining the analysis for both the cases results in a branching vector of $(\frac{3}{2}, \frac{5}{2}, \frac{5}{2}, \frac{3}{2}, 2)$ for the rule **B6**.

Finally, we combine all the above results to obtain the following theorem.

Theorem 2. VERTEX COVER ABOVE LP can be solved in time $O^*((2.3146)^{k-vc^*(G)})$.

Proof. Let us fix $\mu = \mu(G, k) = k - vc^*(G)$. We have thus shown that the preprocessing rules do not increase the measure. Branching rules **B1** or **B2** or **B3** results in a $(1, 1)$ decrease in $\mu(G, k) = \mu$, resulting in the recurrence $T(\mu) \leq T(\mu - 1) + T(\mu - 1)$ which solves to $2^\mu = 2^{k-vc^*(G)}$.

Branching rule **B4** results in a $(\frac{1}{2}, \frac{3}{2})$ decrease in $\mu(G, k) = \mu$, resulting in the recurrence $T(\mu) \leq T(\mu - \frac{1}{2}) + T(\mu - \frac{3}{2})$ which solves to $2.1479^\mu = 2.1479^{k-vc^*(G)}$.

Branching rule **B5** combined with the next step in the algorithm results in a $(1, \frac{3}{2}, \frac{3}{2})$ branching vector, resulting in the recurrence $T(\mu) \leq T(\mu - 1) + 2T(\mu - \frac{3}{2})$ which solves to $2.3146^\mu = 2.3146^{k-vc^*(G)}$.

We analyzed the way algorithm works after an application of branching rule **B6** before Theorem 2. An overview of drop in measure is given in Figure 9.

This leads to a $(\frac{3}{2}, \frac{5}{2}, 2, \frac{3}{2}, \frac{5}{2})$ branching vector, resulting in the recurrence $T(\mu) \leq T(\mu - 1) + 2T(\mu - \frac{3}{2})$ which solves to $2.3146^\mu = 2.3146^{k-vc^*(G)}$.

Thus, we get an $O^*(2.3146^{k-vc^*(G)})$ algorithm for VERTEX COVER ABOVE LP. \square

5 Applications

In this section we give several applications of the algorithm developed for VERTEX COVER ABOVE LP.

5.1 An algorithm for ABOVE GUARANTEE VERTEX COVER

Since the value of the LP relaxation is at least the size of the maximum matching, our algorithm also runs in time $O^*(2.3146^{k-m})$ where k is the size of the minimum vertex cover and m is the size of the maximum matching.

Theorem 3. ABOVE GUARANTEE VERTEX COVER can be solved in time $O^*(2.3146^\ell)$ time, where ℓ is the excess of the minimum vertex cover size above the size of the maximum matching.

Now by the known reductions in [8, 17, 22] (see also Figure 1) we get the following corollary to Theorem 3.

Corollary 2. ALMOST 2-SAT, ALMOST 2-SAT(v), RHORN-BACKDOOR DETECTION SET can be solved in time $O^*(2.3146^k)$, and KVD_{pm} can be solved in time $O^*(2.3146^{\frac{k}{2}}) = O^*(1.5214^k)$.

5.2 Algorithms for ODD CYCLE TRANSVERSAL and SPLIT VERTEX DELETION

We describe a generic algorithm for both ODD CYCLE TRANSVERSAL and SPLIT VERTEX DELETION. Let $X, Y \in \{\text{Clique, Independent Set}\}$. A graph G is called an (X, Y) -graph if its vertices can be partitioned into X and Y . Observe that when X and Y are both *independent set*, this corresponds to a *bipartite graph* and when X is *clique* and Y is *independent set*, this corresponds to a *split graph*. In this section we outline an algorithm that runs in time $O^*(2.3146^k)$ and solves the following problem.

(X, Y)-TRANSVERSAL SET

Instance: An undirected graph G and a positive integer k .

Parameter: k .

Problem: Does G have a vertex subset S of size at most k such that its deletion leaves a (X, Y) -graph?

We solve the (X, Y)-TRANSVERSAL SET problem by using a reduction to AGVC that takes k to k [25]. We give the reduction here for the sake of the completeness. Let $X, Y \in \{\text{Clique, Independent Set}\}$

Construction : Given a graph $G = (V, E)$ and (X, Y) , we construct a graph $H(X, Y) = (V(X, Y), E(X, Y))$ as follows. We take two copies of V as the vertex set of $H(X, Y)$, that is, $V(X, Y) = V_1 \cup V_2$ where $V_i = \{u_i \mid u \in V\}$ for $1 \leq i \leq 2$. The set V_1 corresponds to X and the set V_2 corresponds to Y . The edge set of $H(X, Y)$ depends on X or Y being *clique* or *independent set*. If X is *independent set*, then the graph induced on V_1 is made isomorphic to G , that is, for every edge $(u, v) \in E$ we include the edge (u_1, v_1) in $E(X, Y)$. If X is *clique*, then the graph induced on V_1 is isomorphic to the complement of G , that is, for every non-edge $(u, v) \notin E$ we include an edge (u_1, v_1) in $E(X, Y)$. Hence, if X (respectively Y) is *independent set*, then we make the corresponding $H(X, Y)[V_i]$ isomorphic to the graph G and otherwise, we make $H(X, Y)[V_i]$ isomorphic to the complement of G . Finally, we add the perfect matching $P = \{(u_1, u_2) \mid u \in V\}$ to $E(X, Y)$. This completes the construction of $H(X, Y)$.

We first prove the following lemma, which relates the existence of an (X, Y) -induced subgraph in the graph G to the existence of an independent set in the associated auxiliary graph $H(X, Y)$. We use this lemma to relate (X, Y)-TRANSVERSAL SET to AGVC.

Lemma 19. *Let $X, Y \in \{\text{Clique, Independent Set}\}$ and $G = (V, E)$ be a graph on n vertices. Then, G has an (X, Y) -induced subgraph of size t if and only if $H(X, Y)$ has an independent set of size t .*

Proof. Let $S \subseteq V$ be such that $G[S]$ is an (X, Y) -induced subgraph of size t . Let S be partitioned as S_1 and S_2 such that S_1 is X and S_2 is Y . We also know that $H(X, Y) = (V(X, Y), E(X, Y))$. Consider the image of S_1 in V_1 and S_2 in V_2 . Let the images be S_1^H and S_2^H respectively. We claim that $S_1^H \cup S_2^H$ is an independent set of size t in the graph

$H(X, Y)$. To see that this is indeed the case, it is enough to observe that S_i 's partition S , $H[V_i]$ is a copy of G or a copy of the complement of G based on the nature of X and Y . Furthermore, the only edges between any pair of copies of G or \overline{G} in $H(X, Y)$, are of the form $(u_1, u_2), u \in V$, that is, the matching edges.

Conversely, let K be an independent set in $H(X, Y)$ of size t and let K be decomposed as $K_i, 1 \leq i \leq 2$, where $K_i = K \cap V_i$. Let B_i be the set of vertices of G which correspond to the vertices of K_i , that is, $B_i = \{u \mid u \in V, u_i \in K_i\}$ for $1 \leq i \leq 2$. Observe that, for any $u \in V$, K contains at most one of the two copies of u , that is, $|K \cap \{u_1, u_2\}| \leq 1$ for any $u \in V$. Hence, $|B_1| + |B_2| = t$. Recall that, if X is *independent set*, then $H(X, Y)[V_1]$ is a copy of G and hence B_1 is an independent set in G and if X is *clique*, then $H(X, Y)[V_1]$ is a copy of the complement of G and hence K_1 is an independent set in \overline{G} , and thus B_1 induces a clique in G . The same can be argued for the two cases for Y . Hence, the graph $G[B_1 \cup B_2]$ is indeed an (X, Y) -graph of size t . This completes the proof of the lemma. \square

Using Lemma 19, we obtain the following result.

Lemma 20. *Let $X, Y \in \{\text{Clique}, \text{Independent Set}\}$ and G be a graph on n vertices. Then G has a set of vertices of size at most k whose deletion leaves an (X, Y) -graph if and only if $H(X, Y)$ has a vertex cover of size at most $n + k$, where n is the size of the perfect matching of $H(X, Y)$.*

Proof. By Lemma 19 we have that G has an (X, Y) -induced subgraph of size t if and only if $H(X, Y)$ has an independent set of size t . Thus, G has an (X, Y) -induced subgraph of size $n - k$ if and only if $H(X, Y)$ has an independent set of size $n - k$. But this can happen if and only if $H(X, Y)$ has a vertex cover of size at most $2n - (n - k) = n + k$. This proves the claim. \square

Combining the above lemma with Theorem 3, we have the following.

Theorem 4. (X, Y) -TRANSVERSAL SET can be solved in time $O^*(2.3146^k)$.

As a corollary to the above theorem we get the following new results.

Corollary 3. ODD CYCLE TRANSVERSAL and SPLIT VERTEX DELETION can be solved in time $O^*(2.3146^k)$.

Observe that the reduction from EDGE BIPARTIZATION to ODD CYCLE TRANSVERSAL represented in Figure 1, along with the above corollary implies that EDGE BIPARTIZATION can also be solved in time $O^*(2.3146^k)$. However, we note that Guo et al. [9] have given an algorithm for this problem running in time $O^*(2^k)$.

5.3 An algorithm for KÖNIG VERTEX DELETION

A graph G is called König if the size of a minimum vertex cover equals that of a maximum matching in the graph. Clearly bipartite graphs are König but there are non-bipartite graphs that are König (a triangle with an edge attached to one of its vertices, for example). Thus the KÖNIG VERTEX DELETION problem, as stated below, is closely connected to ODD CYCLE TRANSVERSAL.

KÖNIG VERTEX DELETION (KVD)

Instance: An undirected graph G and a positive integer k .
Parameter: k .
Problem: Does G have a vertex subset S of size at most k such that $G \setminus S$ is a König graph?

If the input graph G to KÖNIG VERTEX DELETION has a perfect matching then this problem is called KVD_{pm} . By Corollary 2, we already know that KVD_{pm} has an algorithm with running time $O^*(1.5214^k)$ by a polynomial time reduction to AGVC, that takes k to $k/2$. However, there is no known reduction if we do not assume that the input graph has a perfect matching and it required several interesting structural theorems in [18] to show that KVD can be solved as fast as AGVC. Here, we outline an algorithm for KVD that runs in $O^*(1.5214^k)$ and uses an interesting reduction rule. However, for our algorithm we take a detour and solve a slightly different, although equally interesting problem. Given a graph, a set S of vertices is called *König vertex deletion set (kvd set)* if its removal leaves a König graph. The auxiliary problem we study is following.

VERTEX COVER PARAM BY KVD

Instance: An undirected graph G , a König vertex deletion set S of size at most k and a positive integer ℓ .
Parameter: k .
Problem: Does G have a vertex cover of size at most ℓ ?

This fits into the recent study of problems parameterized by other structural parameters. See, for example ODD CYCLE TRANSVERSAL parameterized by various structural parameters [12] or TREewidth parameterized by vertex cover [1] or VERTEX COVER parameterized by feedback vertex set [11]. For our proofs we will use the following characterization of König graphs.

Lemma 21. [18, Lemma 1] *A graph $G = (V, E)$ is König if and only if there exists a bipartition of V into $V_1 \uplus V_2$, with V_1 a vertex cover of G such that there exists a matching across the cut (V_1, V_2) saturating every vertex of V_1 .*

Note that in VERTEX COVER PARAM BY KVD, $G \setminus S$ is a König graph. So one could branch on all subsets of S to include in the output vertex cover, and for those elements not picked in S , we could pick its neighbors in $G \setminus S$ and delete them. However, the resulting graph need not be König adding to the complications. Note, however, that such an algorithm would yield an $O^*(2^k)$ algorithm for VERTEX COVER PARAM BY OCT. That is, if S were an odd cycle transversal then the resulting graph after deleting the neighbors of vertices not picked from S will remain a bipartite graph, where an optimum vertex cover can be found in polynomial time.

Given a graph $G = (V, E)$ and two disjoint vertex subsets V_1, V_2 of V , we let (V_1, V_2) denote the bipartite graph with vertex set $V_1 \cup V_2$ and edge set $\{\{u, v\} : \{u, v\} \in E \text{ and } u \in V_1, v \in V_2\}$. Now, we describe an algorithm based on Theorem 1, that solves VERTEX COVER PARAM BY KVD in time $O^*(1.5214^k)$.

Theorem 5. VERTEX COVER PARAM BY KVD *can be solved in time $O^*(1.5214^k)$.*

Proof. Let G be the input graph, S be a kvd set of size at most k . We first apply Lemma 1 on $G = (V, E)$ and obtain an optimum solution to LPVC(G) such that all $\frac{1}{2}$ is the unique optimum solution to LPVC($G[V_{1/2}^x]$). Due to Lemma 2, this implies that there exists a minimum vertex cover of G that contains all the vertices in V_1^x and none of the vertices in V_0^x . Hence, the problem reduces to finding a vertex cover of size $\ell' = \ell - |V_1^x|$ for the graph $G' = G[V_{1/2}^x]$. Before we describe the rest of the algorithm, we prove the following lemma regarding kvd sets in G and G' which shows that if G has a kvd set of size at most k then so does G' . Even though this looks straight forward, the fact that König graphs are not hereditary (i.e. induced subgraphs of König graphs need not be König) makes this a non-trivial claim to prove.

Lemma 22. *Let G and G' be defined as above. Let S be a kvd set of graph G of size at most k . Then, there is a kvd set of graph G' of size at most k .*

Proof. It is known that the sets $(V_0^x, V_1^x, V_{1/2}^x)$ form a *crown decomposition* of the graph G [4]. In other words, $N(V_0^x) = V_1^x$ and there is a matching saturating V_1^x in the bipartite graph (V_1^x, V_0^x) . The set V_0^x is called the *crown* and the set V_1^x is called the *head* of the decomposition. For ease of presentation, we will refer to the set V_0^x as C , V_1^x as H and the set $V_{1/2}^x$ as R . In accordance with Lemma 21, let A be the minimum vertex cover and let I be the corresponding independent set of $G \setminus S$ such that there is a matching saturating A across the bipartite graph (A, I) . First of all, note that if the set S is disjoint from $C \cup H$, $H \subseteq A$, and $C \subseteq I$, we are done, since the set S itself can be taken as a kvd set for G' . This last assertion follows because there exists a matching saturating H into C . Hence, we may assume that this is not the case. However, we will argue that given a kvd set of G of size at most k we will always be able to modify it in a way that it is of size at most k , it is disjoint from $C \cup H$, $H \subseteq A$, and $C \subseteq I$. This will allow us to prove our lemma. Towards this, we now consider the set $H' = H \cap I$ and consider the following two cases.

1. H' is empty. We now consider the set $S' = S \setminus (C \cup H)$ and claim that S' is also a kvd set of G of size at most k such that $G \setminus S'$ has a vertex cover $A' = (A \setminus C) \cup H$ with the corresponding independent set being $I' = I \cup C$. In other words, we move all the vertices of H to A and the vertices of C to I . Clearly, the size of the set S' is at most that of S . The set I' is independent since I was initially independent, and the newly added vertices have edges only to vertices of H , which are not in I' . Hence, the set A' is indeed a vertex cover of $G \setminus S'$. Now, the vertices of R , which lie in A , (and hence A') were saturated by vertices not in H , since $H \cap I$ was empty. Hence, we may retain the matching edges saturating these vertices, and as for the vertices of H , we may use the matching edges given by the crown decomposition to saturate these vertices and thus there is a matching saturating every vertex in A' across the bipartite graph (A', I') . Hence, we now have a kvd set S' disjoint from $C \cup H$, such that H is part of the vertex cover and C lies in the independent set of the König graph $G \setminus S'$.
2. H' is non empty. Let C_1 be the set of vertices in $A \cap C$ which are adjacent to H' (see Fig. 22), let C_2 be the set of vertices in $C \cap S$, which are adjacent to H' , and let P be

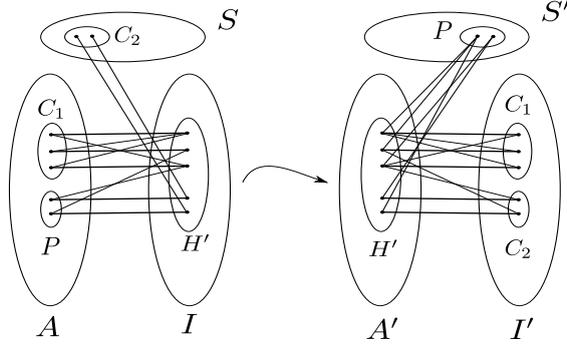


Figure 11: An illustration of case 2 of Lemma 22

the set of vertices of $R \cap A$ which are saturated by vertices of H' in the bipartite graph (A, I) . We now consider the set $S' = (S \setminus C_2) \cup P$ and claim that S' is also a kvd set of G of size at most k such that $G \setminus S'$ has a minimum vertex cover $A' = (A \setminus (C_1 \cup P)) \cup H'$ with the corresponding independent set being $I' = (I \setminus H') \cup (C_1 \cup C_2)$. In other words, we move the set H' to A , the sets C_1 and C_2 to I and the set P to S . The set I' is independent since I was independent and the vertices added to I are adjacent only to vertices of H , which are not in I' . Hence, A' is indeed a vertex cover of $G \setminus S'$. To see that there is still a matching saturating A' into I' , note that any vertex previously saturated by a vertex not in H can still be saturated by the same vertex. As for vertices of H' , which have been newly added to A , they can be saturated by the vertices in $C_1 \cup C_2$. Observe that $C_1 \cup C_2$ is precisely the neighborhood of H' in C and since there is a matching saturating H in the bipartite graph (H, C) by Hall's Matching Theorem we have that for every subset $\hat{H} \subseteq H$, $|N(\hat{H}) \cap (C_1 \cup C_2)| \geq |\hat{H}|$. Hence, by Hall's Matching Theorem there is a matching saturating A' in the bipartite graph (A', I') . It now remains to show that $|S'| \leq k$.

Since $N(H') = C_1 \cup C_2$ in the bipartite graph (C, H) , we know that $|C_1| + |C_2| \geq |H'|$. In addition, the vertices of C_1 have to be saturated in the bipartite graph (A, I) by vertices in H' . Hence, we also have that $|C_1| + |P| \leq |H'|$. This implies that $|C_2| \geq |P|$. Hence, $|S'| \leq |S| \leq k$. This completes the proof of the claim. But now, notice that we have a kvd set of size at most k such that there are no vertices of H in the independent set side of the corresponding König graph. Thus, we have fallen into Case 1, which has been handled above.

This completes the proof of the lemma. \square

We now show that $\mu = vc(G') - vc^*(G') \leq \frac{k}{2}$. Let O be a kvd set of G' and define G'' as the König graph $G' \setminus O$. It is well known that in König graphs, $|M| = vc(G'') = vc^*(G'')$, where M is a maximum matching in the graph G'' . This implies that $vc(G') \leq vc(G'') + |O| = |M| + |O|$. But, we also know that $vc^*(G') \geq |M| + \frac{1}{2}(|O|)$ and hence, $vc(G') - vc^*(G') \leq \frac{1}{2}(|O|)$. By Lemma 22, we know that there is an O such that $|O| \leq k$ and hence, $vc(G') - vc^*(G') \leq \frac{k}{2}$.

By Corollary 1, we can find a minimum vertex cover of G' in time $O^*(2.3146^{vc(G') - vc^*(G')})$

and hence in time $O^*(2.3146^{k/2})$. If the size of the minimum vertex cover obtained for G' is at most ℓ' , then we return yes else we return no. This completes the proof of the theorem. \square

It is known that, given a minimum vertex cover, a minimum sized kvd set can be computed in polynomial time [18]. Hence, Theorem 5 has the following corollary.

Corollary 4. *KVD can be solved in time $O^*(1.5214^k)$.*

Since the size of a minimum Odd Cycle Transversal is at least the size of a minimum Konig Vertex Deletion set, we also have the following corollary.

Corollary 5. *VERTEX COVER PARAM BY OCT can be solved in time $O^*(1.5214^k)$.*

5.4 A simple improved kernel for VERTEX COVER

We give a kernelization for VERTEX COVER based on Theorem 1 as follows. Exhaustively, apply the Preprocessing rules 1 through 3 (see Section 3). When the rules no longer apply, if $k - vc^*(G) \leq \log k$, then solve the problem in time $O^*(2.3146^{\log k}) = O(n^{O(1)})$. Otherwise, just return the instance. We claim that the number of vertices in the returned instance is at most $2k - 2 \log k$. Since $k - vc^*(G) > \log k$, $vc^*(G)$ is upper bounded by $k - \log k$. But, we also know that when Preprocessing Rule 1 is no longer applicable, all $\frac{1}{2}$ is the unique optimum to LPVC(G) and hence, the number of vertices in the graph G is twice the value of the optimum value of LPVC(G). Hence, $|V| = 2vc^*(G) \leq 2(k - \log k)$. Observe that by the same method we can also show that in the reduced instance the number of vertices is upper bounded by $2k - c \log k$ for any fixed constant c . Independently, Lampis [14] has also shown an upper bound of $2k - c \log k$ on the size of a kernel for VERTEX COVER for any fixed constant c .

6 Conclusion

We have demonstrated that using the drop in LP values to analyze in branching algorithms can give powerful results for parameterized complexity. We believe that our algorithm is the beginning of a race to improve the running time bound for AGVC and possibly for the classical VERTEX COVER problem, for which there has been no progress in the last several years after an initial plethora of results.

Our other contribution is to exhibit several parameterized problems that are equivalent to or reduce to AGVC through parameterized reductions. We observe that as the parameter change in these reductions are linear, any upper or lower bound results for kernels for one problem will carry over for the other problems too (subject to the directions of the reductions). For instance, recently, Kratsch and Wahlström [13] studied the kernelization complexity of AGVC and obtained a randomized polynomial sized kernel for it through matroid based techniques. This implies a randomized polynomial kernel for all the problems in this paper.

References

- [1] H. L. BODLAENDER, B. M. P. JANSEN, AND S. KRATSCH, *Preprocessing for treewidth: A combinatorial analysis through kernelization*, in ICALP (1), L. Aceto, M. Henzinger, and J. Sgall, eds., vol. 6755 of Lecture Notes in Computer Science, Springer, 2011, pp. 437–448.
- [2] L. CAI, *Fixed-parameter tractability of graph modification problems for hereditary properties*, Inf. Process. Lett., 58 (1996), pp. 171–176.
- [3] J. CHEN, I. A. KANJ, AND G. XIA, *Improved upper bounds for vertex cover*, Theor. Comput. Sci., 411 (2010), pp. 3736–3756.
- [4] M. CHLEBÍK AND J. CHLEBÍKOVÁ, *Crown reductions for the minimum weighted vertex cover problem*, Discrete Applied Mathematics, 156 (2008), pp. 292–312.
- [5] M. CYGAN, M. PILIPCZUK, M. PILIPCZUK, AND J. O. WOJTASZCZYK, *On multiway cut parameterized above lower bounds*, in IPEC, 2011.
- [6] R. G. DOWNEY AND M. R. FELLOWS, *Parameterized Complexity*, Springer-Verlag, New York, 1999.
- [7] J. FLUM AND M. GROHE, *Parameterized Complexity Theory*, Texts in Theoretical Computer Science. An EATCS Series, Springer-Verlag, Berlin, 2006.
- [8] G. GOTTLOB AND S. SZEIDER, *Fixed-parameter algorithms for artificial intelligence, constraint satisfaction and database problems*, Comput. J., 51 (2008), pp. 303–325.
- [9] J. GUO, J. GRAMM, F. HÜFFNER, R. NIEDERMEIER, AND S. WERNICKE, *Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization*, J. Comput. Syst. Sci., 72 (2006), pp. 1386–1396.
- [10] F. HÜFFNER, *Algorithm engineering for optimal graph bipartization*, J. Graph Algorithms Appl., 13 (2009), pp. 77–98.
- [11] B. M. P. JANSEN AND H. L. BODLAENDER, *Vertex cover kernelization revisited: Upper and lower bounds for a refined parameter*, in STACS, T. Schwentick and C. Dürr, eds., vol. 9 of LIPIcs, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011, pp. 177–188.
- [12] B. M. P. JANSEN AND S. KRATSCH, *On polynomial kernels for structural parameterizations of odd cycle transversal*, in IPEC, 2011.
- [13] S. KRATSCH AND M. WAHLSTRÖM, *Representative sets and irrelevant vertices: New tools for kernelization*, CoRR, abs/1111.2195 (2011).
- [14] M. LAMPIS, *A kernel of order $2k - c \log k$ for vertex cover*, Inf. Process. Lett., 111 (2011), pp. 1089–1091.
- [15] D. LOKSHTANOV, S. SAURABH, AND S. SIKDAR, *Simpler parameterized algorithm for oct*, in IWOCA, J. Fiala, J. Kratochvíl, and M. Miller, eds., vol. 5874 of Lecture Notes in Computer Science, Springer, 2009, pp. 380–384.

- [16] M. MAHAJAN AND V. RAMAN, *Parameterizing above guaranteed values: Maxsat and maxcut*, J. Algorithms, 31 (1999), pp. 335–354.
- [17] D. MARX AND I. RAZGON, *Constant ratio fixed-parameter approximation of the edge multicut problem*, Inf. Process. Lett., 109 (2009), pp. 1161–1166.
- [18] S. MISHRA, V. RAMAN, S. SAURABH, S. SIKDAR, AND C. R. SUBRAMANIAN, *The complexity of könig subgraph problems and above-guarantee vertex cover*, Algorithmica, 58 (2010).
- [19] G. L. NEMHAUSER AND L. E. TROTTER, *Properties of vertex packing and independence system polyhedra*, Mathematical Programming, 6 (1974), pp. 48–61. 10.1007/BF01580222.
- [20] ———, *Vertex packings: Structural properties and algorithms*, Mathematical Programming, 8 (1975), pp. 232–248. 10.1007/BF01580444.
- [21] J.-C. PICARD AND M. QUEYRANNE, *On the integer-valued variables in the linear vertex packing problem*, Mathematical Programming, 12 (1977), pp. 97–101.
- [22] V. RAMAN, M. S. RAMANUJAN, AND S. SAURABH, *Paths, flowers and vertex cover*, in Algorithms – ESA 2011, C. Demetrescu and M. Halldórsson, eds., vol. 6942 of Lecture Notes in Computer Science, Springer Berlin / Heidelberg, 2011, pp. 382–393.
- [23] I. RAZGON AND B. O’SULLIVAN, *Almost 2-sat is fixed-parameter tractable.*, J. Comput. Syst. Sci., 75 (2009), pp. 435–450.
- [24] B. A. REED, K. SMITH, AND A. VETTA, *Finding odd cycle transversals*, Oper. Res. Lett., 32 (2004), pp. 299–301.
- [25] S. SAURABH, *Exact Algorithms for some parameterized and optimization problems on graphs*, PhD thesis, 2008.

7 Appendix: Problem Definitions

VERTEX COVER

Instance: An undirected graph G and a positive integer k .
Parameter: k .
Problem: Does G have a vertex cover of of size at most k ?

ABOVE GUARANTEE VERTEX COVER (AGVC)

Instance: An undirected graph G , a maximum matching M and a positive integer ℓ .
Parameter: ℓ .
Problem: Does G have a vertex cover of of size at most $|M| + \ell$?

VERTEX COVER ABOVE LP

Instance: An undirected graph G , positive integers k and $\lceil vc^*(G) \rceil$, where $vc^*(G)$ is the minimum value of LPVC.
Parameter: $k - \lceil vc^*(G) \rceil$.
Problem: Does G have a vertex cover of of size at most k ?

A graph G is called an *bipartite* if its vertices can be partitioned into X and Y such that X and Y are independent sets.

ODD CYCLE TRANSVERAL (OCT)

Instance: An undirected graph G and a positive integer k .
Parameter: k .
Problem: Does G have a vertex subset S of size at most k such that $G \setminus S$ is a bipartite graph?

EDGE BIPARTIZATION (EB)

Instance: An undirected graph G and a positive integer k .
Parameter: k .
Problem: Does G have an edge subset S of size at most k such that $G' = (V, E \setminus S)$ is a bipartite graph?

A graph G is called an *split* if its vertices can be partitioned into X and Y such that X is a clique and Y is an independent set.

SPLIT VERTEX DELETION

Instance: An undirected graph G and a positive integer k .
Parameter: k .
Problem: Does G have a vertex subset S of size at most k such that $G \setminus S$ is a split graph?

A graph G is called an *König* if the size of a maximum matching is equal to the size of a minimum vertex cover.

KÖNIG VERTEX DELETION (KVD)

Instance: An undirected graph G and a positive integer k .
Parameter: k .
Problem: Does G have a vertex subset S of size at most k such that $G \setminus S$ is a König graph?

If the input graph to KVD has a perfect matching then we call it KVD_{pm} .

Given a 2-SAT formula ϕ on variables x_1, \dots, x_n , and with clauses C_1, \dots, C_m , we define *deleting* a clause from ϕ as removing the clause from the formula ϕ and deleting a variable from ϕ as removing all the clauses which involve that variable, from ϕ .

ALMOST 2-SAT

Instance: A 2-SAT formula ϕ and a positive integer k .
Parameter: k .
Problem: Does there exist a set of at most k clauses, whose deletion from ϕ makes the resulting formula satisfiable?

ALMOST 2-SAT-VARIABLE VERSION (ALMOST 2-SAT(v))

Instance: A 2-SAT formula ϕ and a positive integer k .
Parameter: k .
Problem: Does there exist a set of at most k variables, whose deletion from ϕ makes the resulting formula satisfiable?

Given a graph G , a vertex subset K of G is said to be a König vertex deletion (KVD) set if the graph $G \setminus K$ is a König graph.

VERTEX COVER PARAM BY KVD

Instance: An undirected graph G , a positive integer k , and a set K , which is a KVD set for G .
Parameter: $|K|$.
Problem: Does G have a vertex cover of size at most k ?

VERTEX COVER PARAM BY OCT

Instance: An undirected graph G , a positive integer k , and a set K , which is an OCT for G .
Parameter: $|K|$.
Problem: Does G have a vertex cover of size at most k ?

HORN denotes the set of CNF formulas where each clause contains at most one positive literal. RHORN denotes the class of renamable HORN CNF formulas, that is, of CNF formulas F for which there exists a set $X \subset \text{var}(F)$ such that, replacing in the clauses of F the literal x by \bar{x} and the literal \bar{x} by x whenever $x \in X$, yields a Horn formula. The set $\text{var}(F)$ contains the variables contained in F . Obviously, RHORN properly contains HORN. For a CNF formula F and a set of variables $B \subseteq \text{var}(F)$ let $F \setminus B$ denote the CNF formula $\{C \setminus (B \cup \bar{B}) : C \in F\}$, that is, set of clauses obtained after deleting the variables and its negation in the set B . For a formula F , we say that a set $B \subseteq \text{var}(F)$ is *deletion RHORN-backdoor set* if $F \setminus B$ is in RHORN.

RHORN-BACKDOOR DETECTION SET (RHBDS)

Instance: A CNF formula ϕ and a positive integer k .

Parameter: k .

Problem: Does there exists a deletion RHORN-backdoor set of size at most k ?